State of Practical Applicability of Regression Testing Research: A Live Systematic Literature Review

RENAN GRECA, Gran Sasso Science Institute, Italy and ISTI-CNR, Italy
BRENO MIRANDA, Universidade Federal de Pernambuco, Brazil
ANTONIA BERTOLINO, ISTI-CNR, Italy

Context: Software regression testing refers to rerunning test cases after the system under test is modified, ascertaining that the changes have not (re-)introduced failures. Not all researchers' approaches consider applicability and scalability concerns, and not many have produced an impact in practice. **Objective:** One goal is to investigate industrial relevance and applicability of proposed approaches. Another is providing a live review, open to continuous updates by the community. **Method:** A systematic review of regression testing studies that are clearly motivated by or validated against industrial relevance and applicability is conducted. It is complemented by follow-up surveys with authors of the selected papers and 23 practitioners. **Results:** A set of 79 primary studies published between 2016-2022 is collected and classified according to approaches and metrics. Aspects relative to their relevance and impact are discussed, also based on their authors' feedback. All the data are made available from the live repository that accompanies the study. **Conclusions:** While widely motivated by industrial relevance and applicability, not many approaches are evaluated in industrial or large-scale open-source systems, and even fewer approaches have been adopted in practice. Some challenges hindering the implementation of relevant approaches are synthesized, also based on the practitioners' feedback.

1 INTRODUCTION

Regression testing (RT) consists of rerunning the previously executed tests when the software under test (SUT) evolves to verify that no new failures, or regressions, have been introduced. RT is quite a prominent problem in industry that can draw a large part of the testing budget [51, 60]. The reason is that, as development proceeds, the test suite size tends to grow significantly, making the re-execution of all test cases (the *retest all* strategy) impractical. Besides, with the growing adoption of short release cycles and Continuous Integration practices, the role of RT becomes ever more central [74, 105].

As a consequence, RT is a very active research topic. Since the '80s [63, 121], many approaches have been investigated for making RT more effective and efficient, including different techniques: *test case prioritization* (TCP) [56] determines an execution ordering that ideally gives precedence to the most effective test cases; *test case selection* (TCS) [55] takes a sample of test cases for execution generally based on the recently introduced changes; *test suite reduction* (TSR) [95], also referred to as test suite minimization [121], aims at removing redundant test cases according to some criterion, for instance code or requirements coverage; and *test suite augmentation* (TSA) [101] supports the creation of new test cases, if needed, for testing the changed program behavior.

Unfortunately, the research community efforts over the years to mitigate RT cost and complexity do not seem to have produced the desired impact. A 2010 study [33] aiming at understanding RT practice already highlighted several divergences between software testing research and practice, notwithstanding in 2017 Garousi and Felderer

Authors' addresses: Renan Greca, renan.greca@gssi.it, Gran Sasso Science Institute, L'Aquila, Italy and ISTI-CNR, Pisa, Italy; Breno Miranda, bafm@cin.ufpe.br, Universidade Federal de Pernambuco, Recife, Brazil; Antonia Bertolino, antonia.bertolino@isti.cnr.it, ISTI-CNR, Pisa, Italy.

0360-0300/2023/1-ART $15.00
https://doi.org/10.1145/3579851

[39] still called them as "*worlds apart*". Indeed, in a recent systematic review of the RT literature aiming at identifying approaches with *industrial relevance and applicability* (henceforth referred to as IR&A), bin Ali et al. [11] could only select 38 primary studies out of an initial pool of 1068 collected works. In other words, their study would imply that less than 4% of the published works on RT could be of interest to industry.

It is disappointing that so many solutions proposed by the research to improve RT cost/effectiveness do not find their way into practice. In this study we provide a Systematic Literature Review (SLR) with the purpose of reviewing the IR&A of RT approaches published in the latest years, i.e., since 2016. For the sake of comprehensiveness, we characterize as having IR&A not only those studies that report an evaluation on industrial applications (as was done by bin Ali et al. [11]), but also approaches that are explicitly motivated by industrial problems, or by related concerns, such as costs, scalability, or impact on the development procedures. Within this scope, we performed a systematic search over the five main digital libraries (ACM, IEEE, Springer, Scopus and Wiley) for RT studies mentioning industry or practice or applicability or scalability (or similar wordings) in their abstract, and completed this search with a snowballing cycle. We collected 1320 candidate papers published between 2016 and 2022 (780 via query, 540 via snowballing), and after applying a systematic selection process we identified a total of 78 primary studies that present IR&A approaches. However, we understand that there is not a direct mapping between motivation and results, and approaches stemming from applicability concerns could end up with having low significance. In order to get a better assessment of the long-term impact of the papers after publication, we complemented our literature review with a survey sent to the authors of all the selected studies, asking them about the outcome of their research post-publication. We received responses from authors of 64% of the papers, reporting both positive and negative outcomes, including some of the reasons why an approach was successful or unsuccessful after attempts of implementation. Some of the authors also signaled additional papers for consideration of the review, out of which we selected only 1.

Our review includes a total of 79 primary studies. Based on a full reading of the selected papers and on the feedback received by their authors, we discuss the main characteristics of IR&A approaches, how they tackle applicability concerns, and whether they produced in impact in practice and why (or why not). We then also conducted a further survey among test practitioners to get their opinion in order to comment and possibly validate our conclusions. By applying a convenience sampling method we got answers from 23 practitioners who confirmed our findings and provided further useful insights in our study of investigating IR&A of proposed RT approaches.

Finally, as the paper title indicates, this review is conceived as a *live* systematic review[1]. In our analysis of recent secondary studies, we noticed that a gap of one or even two years elapses between the covered period of literature and the year the review appears. This is understandable because the authors may need substantial time for analyzing the selected studies and then writing the article, and then several months will typically be taken for the peer reviewing process. Moreover, even if this temporal gap is reduced to a minimum, as long as the investigated topic remains active, new primary studies will always appear, soon distancing any SLR from the status of literature. If the purpose of a SLR is to provide an up-to-date summary of existing work on a topic, frequent updates are required to include newly appearing relevant studies. Also, it can happen that the original research questions and findings lose relevance, or are superseded by newer results. As previously questioned in other disciplines [38], Mendes et al. [75] have recently investigated the issue of SLRs in SE becoming obsolete and of when and how they would need to be updated. Specifically, in line with [38], they conclude that SLRs should be updated based on two conditions: *i)* new relevant methods, studies or information become available; or *ii)* the adoption or inclusion of previous and new research cause an impact to the findings, conclusions or credibility of the original SLR. In consideration

[1]We notice that our concept of a "live" systematic review, while inspired by similar aims, should not be confused with the much more formalized approach for conducting *living systematic reviews* recently adopted in medicine, as illustrated by https://community.cochrane.org/review-production/production-resources/living-systematic-reviews.

of these challenges, rather than providing a static repository of the studies found while conducting the review, together with the article we release an *open and updatable repository*, which is an integral part of this review work.

The intent is to invite the community to contribute with newly published works that propose or discuss approaches to RT satisfying IR&A, or even with works published in the period we cover but that for some reason escaped our selection. This will allow us not only to keep track of newly published studies, but also to recover papers with purely theoretical motivations that eventually provided benefits in practice. At periodic intervals (planned to be once per year), we will re-run the queries and check suggestions from the community to decide how to update the collection of studies.

In conclusion, this work provides the following contributions:

- A systematic literature review of recent RT techniques emphasizing IR&A;
- A survey among the authors of the selected primary studies that provides information about their impact;
- A survey among practitioners to validate and complement our findings;
- A live repository of primary studies in IR&A of RT techniques, allowing for continuous and periodic updates of this SLR[2].

The paper is structured as follows: in the next section we overview related reviews, and in Section 3 we describe the study methodology, including the three Research Questions (RQs) we formulate, the selection and data extraction process and the authors' survey. We then answer the RQs: in Section 4 we answer RQ1, by overviewing and classifying existing techniques; in Section 5 we answer RQ2, which addresses IR&A concerns; in Section 6 we answer RQ3, about the impact obtained by the collected studies. In Section 7 we discuss threats to the validity of the study, and in Section 8 we present the live repository. Finally, in Section 9, we conclude the paper with a list of challenges identified in the study, including some suggestions of how to handle them, which may serve as future research directions.

2 EXISTING SECONDARY STUDIES

The first recommended step before undertaking any new systematic review is to verify that such a study is actually needed [58]. Indeed, in view of the large set of papers published every year on RT techniques and related topics, it is natural that a good number of secondary studies reviewing the regression testing literature has also been produced.

The already cited study by bin Ali et al. [11] has previously verified whether existing reviews of literature regarding RT techniques took into consideration IR&A. After a systematic search, they found eleven secondary studies spanning over 2008-2017, including [15, 16, 34, 36, 47, 55, 92, 97, 106, 121, 124]. After a thorough analysis of those reviews they concluded that at the time none of them addressed satisfactorily such aspects. The authors hence used such studies[3] as the start-set for a snowball sampling search, launched in August 2016. In order to verify if another review is needed, it is hence necessary to conduct a thorough examination of existing secondary studies on RT published since 2016.

We performed a search for secondary studies on RT over the same libraries queried for the primary studies (see Section 3.3) and complemented the search results with a snowballing cycle. We eventually identified 22 works published since 2016 that are listed in the first column of Table 1, whereas the second column includes the year the review was published.

In the third column of Table 1 we report which RT techniques are covered in the study. Most reviews only focus on TCP approaches [2, 3, 9, 46, 48, 49, 56, 66, 77, 78, 91, 100]. One study is dedicated solely to TCS [55], one other study to TSR [95], and again only one to TSA [26]. Finally, seven secondary studies investigate primary studies on multiple RT techniques [8, 11, 27, 87, 97–99].

[2]Available at: https://renangreca.github.io/literature-repository/.

[3]Actually 10 of them, as the authors explain that the 2017 survey [55] only appeared after they had concluded the analysis.

Paper	Year	Techniques	Nº	Period	Systematic?	Context
Do [27]	2016	TCP, TCS, TSR	12	2010-2016		
Hao et al. [46]	2016	TCP	27	2010-2016		
Rosero et al. [97]	2016	TCP, TCS, TSR	25	2000-2014		
Kazmi et al. [55]	2017	TCS	47	2007-2015	✓	
Bajaj and Sangwan [8]	2018	TCP, TCS, TSR	15	1999-2016		Nature-inspired
Khatibsyarbini et al. [56]	2018	TCP	80	1999-2016	✓	
Mukherjee and Patnaik [78]	2018	TCP	90	2001-2018	✓	
Rehman Khan et al. [95]	2018	TSR	113	1993-2016	✓	
Bajaj and Sangwan [9]	2019	TCP	20	2006-2018	✓	Genetic
bin Ali et al. [11]	2019	TCP, TCS, TSR	38	2002-2017	Mix	
Danglot et al. [26]	2019	TSA	49[1]	1993-2017	✓	
Lou et al. [66]	2019	TCP	191	1997-2016	✓	
Hasnain et al. [48]	2020	TCP	65	2001-2017	✓	Web services
Prado Lima and Vergilio [91]	2020	TCP	35	2009-2019	✓	Continuous integration
Abdul Manan et al. [2]	2021	TCP	20	2011-2020	✓	Combinatorial
Hasnain et al. [49]	2021	TCP	24	2007-2019	✓	Ontology-based
Mohd-Shafie et al. [77]	2021	TCP	22[2]	2005-2018	✓	Model-based
Rosero et al. [98]	2021	TCP, TCS, TSR	40	2002-2020	✓	
Samad et al. [100]	2021	TCP	52	2007-2019	✓	
Ahmed et al. [3]	2022	TCP	21	2001-2019	✓	Value-based
Pan et al. [87]	2022	TCP, TCS	29	2006-2020	✓	Machine learning
Sadri-Moshkenani et al. [99]	2022	TCP, TCS, TSR	13[2]	2015-2019	✓	Cyber-physical

1: Ref. [26] covers test amplification, which is a wider scope than test augmentation, and the reported number of 49 primary studies includes the whole field. 2.: For Refs. [77] and [99] the reported number of primary studies also includes papers addressing test generation.

Table 1. Overview of existing secondary studies on RT.

In the 4th and 5th columns, we report the number of primary studies reviewed and the interval of years to which they belong, whereas the 6th column is checked if the review is conducted in systematic way. Finally, in the last column, we also report on the context of the review, if it focuses on techniques using a specific approach or covers a specific application domain. A version of this data is also included in our online repository (Section 8), with some additional notes; the intent is to update the table as more secondary studies are written and published.

For the sake of comparison, in the following paragraphs we briefly report the motivations behind the 22 reviews, grouped by the targeted technique (i.e., TCP, TCS, TSR, TSA, or multiple techniques).

TCP only. The work by Hao et al. [46] aims at reviewing the advancements in TCP and identifying open challenges. Similar goals are pursued by Lou et al. [66], who analyze the primary studies along six aspects: algorithms, criteria, measurements, constraints, empirical studies, and scenarios. The objective of Khatibsyarbini et al. [56] was to review the experimental evidence relative to the most recent TCP approaches along with the metrics used for evaluating them. Mukherjee and Patnaik [78] generically aim to identify the most popular and useful TCP approaches. The review by Samad et al. [100] classifies existing work according to the algorithms or models adopted, the subjects of evaluation and the prioritization measures. A number of reviews focus on TCP for specific test approaches, namely: Bajaj and Sangwan [9] cover genetic algorithms; Abdul Manan et al. [2] address combinatorial testing; Hasnain et al. [49] consider ontology-based test methods; Mohd-Shafie et al. [77] cover model-based testing approaches, and Ahmed et al. [3] review TCP techniques that integrate value consideration, either in terms of fault severity or test case cost. Finally Hasnain et al. [48] investigate TCP approaches for web services, whereas Prado Lima and Vergilio [91] study how TCP has been adapted for Continuous Integration environments.

TCS only. The only secondary study focusing on TCS work is Kazmi et al. [55], which aims at presenting the state-of-the-art in effective regression test case selection techniques.

TSR only. The systematic review by Rehman Khan et al. [95] is motivated by the quality assessment of empirical studies employed to evaluate the test reduction approaches.

TSA only. Danglot et al. [26] present the first review on test *amplification*, a novel term they introduce as an umbrella for various activities that aims at improving an existing test suite, including test augmentation, optimization, enrichment, or refactoring. The review is not specifically devoted to RT, but a subset of the primary studies they overview deals with creating new tests for assessing the effects of changes.

Multiple techniques. Among the secondary studies that focus on multiple RT techniques, both Do [27] and Rosero et al. [97] aimed at generically providing an overview of recent research advances. Some authors instead were motivated to study more specific type of techniques: Bajaj and Sangwan [8] aimed at reviewing RT approaches leveraging nature-inspired algorithms, while Pan et al. [87] analyzed TCP and TCS studies that use Machine-Learning based techniques. The review by Sadri-Moshkenani et al. [99] characterizes the approaches and the open challenges for the generation, selection and prioritization of test cases for cyber-physical systems. Rosero et al. [98] provide a preliminary brief mapping of primary studies that report about industrial usage of RT techniques. Finally, the already mentioned study by bin Ali et al. [11] surveys RT research that has industrial relevance and applicability, and also creates a taxonomy useful for the communication between academia and industry.

Nearly all of the secondary studies express some concerns over IR&A of RT techniques, although in most cases these concerns are only mentioned in passing, and are not central to their motivations. Rosero et al. [97] report that only 16% of the surveyed primary studies experimented in industrial context. On the positive side, Do [27] observes that recently, more research is focusing on industrial software or open-source programs of different types. The review by Lou et al. [66] contains a subsection titled "Practical Values" in which they suggest researchers to consider TCP in practical scenarios and to develop usable TCP tools. The only two reviews that specifically target IR&A as this study are: *i)* the aforementioned work by bin Ali et al. [11]. However it selects only papers that performed evaluations with industrial subjects, and was motivated mainly by the goal of establishing a taxonomy for communicating RT research in a way that is accessible and relevant for practitioners; and *ii)* the preliminary work by Rosero et al. [98], but this is is a brief report that just classifies 40 selected primary studies found by searching the TCP literature for the term "industrial" without investigating in depth their characteristics and actual impact.

Considering the list of related secondary studies in Table 1, we conclude that a new secondary study specifically addressing progresses in latest years about IR&A is needed and can provide value to the research and development communities.

3 METHODOLOGY

This section elaborates the entire review process, from its conceptual phases to the list of selected papers and how we organized their contents. First, we establish the research questions that drive both the selection of papers and the data presented and discussed in subsequent sections. Then, we explain the planning and design phase of the survey, followed by its actual execution. We also highlight the data that was extracted from each paper and the process of sending complementary questions to authors via e-mail. We present the survey with practitioners that we conducted in support of the study conclusions. Finally, we make available the relevant data needed to replicate this process, to the extent of possibility.

3.1 Research Questions

With these research questions, we aim to synthesize the current state of RT research in terms of most frequently used approaches and metrics as well as understand researchers' motivations and efforts regarding the practical implementation of their proposed techniques.

RQ1: What are the main approaches used for RT techniques and what are the metrics used to evaluate them? — We want to have an overview of the approaches most used in this field, including what information is required from the software, what algorithms are put into use and what goal are they trying to achieve. In addition, we want to know what metrics are widely accepted among researchers to evaluate such approaches and whether there is evidence to suggest that these metrics correlate to the technique's practical applicability.

RQ2: Is IR&A an explicit concern in RT research? — We want to find out if there is a meaningful number of papers that state IR&A as their motivation and include it as part of their problem description. Additionally, we want to understand what are the main steps that authors usually take towards addressing these concerns with the tools, techniques and solutions they provide.

RQ3: Is there evidence that techniques developed in academia make their way into software in practice? — In an effort to measure the extent of active industry-academia collaborations, we want to highlight examples of techniques that have been put into practice at some point during the development of the work. To provide a clearer picture, we asked authors of the selected studies to provide details of the state of their research post publication.

3.2 Planning and Design of the Review

To answer the questions above, we designed the following literature review process. For the purposes of this review, a "regression testing technique" addresses test case prioritization (TCP), test case selection (TCS), test suite reduction (TSR), or test suite augmentation (TSA). Only papers concerning one or more of these four challenges should be considered. Due to the scale of the available literature and our focused interest in recent developments, we only look for papers published between January/2016 and July/2022. We also only consider papers written in English.

We want to focus on papers that either signify an advance of the state of the art in academia towards practicality, or includes data and discussions that might help guide future researchers to make their research more valuable for practitioners. Thus, to be included in the review, a paper must satisfy at least one of these inclusion criteria:

- It introduces a new regression testing technique and provides evidence that it addresses a real-world concern, or provides substantial benefits in experiments performed with real software;
- It introduces and/or discusses a metric for evaluating regression testing techniques with evidence that it might be valuable in practice; or
- It provides a case study of how regression testing is done in a certain team or company, and provides some insight into the actual needs of practitioners.

We also want to avoid certain topics that are related to regression testing but would increase the scope of the review beyond necessary. A paper should be excluded according to following criteria:

- It is regarding software testing education, as this is a completely different challenge;
- It proposes a technique for test suite generation, which is related albeit distinct problem from TSA[4];
- It is concerning security testing, because it typically requires specific types of techniques [35]; or
- It is concerning software product lines or highly configurable software, as these also present quite different challenges from typical regression testing [29].

After collecting the unfiltered set of papers, the inclusion and exclusion criteria are applied by each author to a random sample set based on their titles, abstracts and, if needed, superficial analysis of the text. In case of divergences in the analysis, the authors should discuss their conclusions. The Cohen Kappa agreement measure [23], a scale from -1 to 1, is used to determine if both authors are generally in agreement regarding this sample of papers. Upon establishing a satisfying agreement value, the analysis of remaining papers are split among the

[4]The primary difference is that *test suite augmentation* presupposes the existence of a test suite to enhance, while *test suite generation* can create a new test suite from scratch.

authors. If it is not clear whether a paper fully satisfies the criteria, it is brought for discussion among all authors until a mutual decision is reached.

After this initial analysis, full-text assessment of the remaining literature is performed. The following quality criteria are to be used to further narrow down the papers that are relevant to our research goals:

- The writing and presentation quality should not hinder comprehension;
- A paper should provide evidence that they address a problem found in real-world software development and/or that the technique was evaluated on real-world software;
- The metrics used for evaluation should be clear and the authors should provide some reasoning as to why they are relevant; and
- In case there are multiple papers by the same group of authors that reference versions of the same work, we keep the most extensive one, avoiding, for instance, a conference paper and a journal paper that address the same research (if they are equivalent, we keep the most recent one).

The results from our queries are complemented by both forward and backward snowballing to improve the comprehensiveness of the review. The same date restrictions and criteria apply to papers found via snowballing.

Finally, a questionnaire with authors (Section 3.5) is used in order to clarify and update some details regarding the selected papers. The authors have the option of suggesting additional papers for consideration in this study; in that case, they should also be analyzed according to the established criteria.

3.3 Executing the Review

We began by assembling a list of keywords that form the basis of our queries, including potential variants of the same terms. These are: test/testing, evaluate/evaluation, metric, indicator, investment, cost, relevant/relevance, industry/industrial, practice/practical/practitioner, applicable/applicability, scale/scalability, regression, selection, prioriti[s/z]ation, amplification/augmentation, reduction/minimi[s/z]ation software. These keywords were used to manually experiment with the ACM and IEEE digital libraries, in order to have a general understanding of the relevance of the results. We found, for example, that the term "regression" would often bring papers on the broad topic of machine learning (even not related to RT), so we had to make sure the word "software" was also mentioned in the abstract.

Once the desired keywords were established, we built a query combining them. The query went through several iterations, in order to maximize the likelihood of finding all the papers that are relevant to our research, while also minimizing the number of papers in excess. The final query was structured as:

```
Title:(test OR testing) AND
Abstract:(evaluat* OR metric OR indicator OR investment OR cost OR relevan*) AND
Abstract:(industr* OR practic* OR applicab* OR scal*) AND
Abstract:(regression OR selection OR prioriti* OR augmentation OR
          amplification OR reduction OR minimi*) AND
Abstract:(software)
```

Queries were executed on five digital libraries: ACM, IEEE, Springer, Scopus and Wiley. The searches were performed on Nov. 4, 2021 and Jul. 27, 2022. Each of the five search engines uses a different syntax for queries, so we adapted the query to each syntax while keeping its overall meaning as similar as possible. We also attempted to include results from Science Direct into the study, but its search engine cannot handle all of the query details. In all of the search engines, we narrowed the results to papers published since January of 2016 and under the fields of Computer Science and Software Engineering. The numbers of results were: ACM (217), IEEE (189), Springer (202), Scopus (285), Wiley (31). Removing exact duplicates that were found in more than one digital library, the initial number of papers considered for the review was 780.

The selection process is illustrated in Figure 1. From the query results, we assembled a spreadsheet with the year, author list, title, abstract and keywords of each paper in a shuffled order, to be systematically screened by two of the authors of this study. Only when needed, we would consult the full text of the paper to ensure topic relevance and the satisfaction of exclusion and quality criteria. A sample of 40 papers was used to calculate the Cohen Kappa measure and establish a consensus. From these, we achieved an agreement value of 0.89, which is considered very high, so we were satisfied with the criteria and the authors' interpretations of them. We split the remaining papers among us for processing and, when uncertain, we would discuss the inclusion of papers together, ultimately deciding to be overly inclusive at this step, and leaving the most rigorous filtering for later. Papers from the same groups of authors were also flagged to then determine if they were describing the same or similar work. Before any snowballing, we detected 86 candidate papers.

Snowballing upon the selected primary studies was performed on Nov. 15 2021 and Aug. 15 2022, using the papers' own references for backward snowballing and Google Scholar for forward snowballing. This resulted in a further 540 papers published since 2016, after removing duplicates of papers already found in the previous review step. The same set of steps described for the query results were applied to the snowballing set, resulting in an additional 108 candidate papers. Combined with the previously-detected candidates, a pool of 194 papers was formed for deeper analysis.

We performed comprehensive full-text analysis of these 194 papers, carefully extracting the information pointed out in Section 3.4 and using that to form the decision of whether or not the paper satisfied our inclusion and quality criteria. Again during this step, we divided the papers among the authors and, in case there was uncertainty regarding one paper, we made the decision together.

Later, when we received responses from the authors of the selected studies (Section 3.5), four papers were brought to our attention. We applied all our aforementioned criteria to these suggestions and decided to incorporate one of them into the review. It had not been caught by either the query or the snowballing process, but we understand that a single missed paper is evidence that our review process has been sufficiently comprehensive.

Finally, our survey, as is presented in this study, contains the 79 papers listed in Table 2: 46 found by the query; 16 from backward snowballing; 16 from forward snowballing; and 1 author suggestion. As is later discussed in Section 8, this group of papers is just the initial contents of the live repository that is made available online. Over time, through updates to the review and submissions by authors, we expect this list to grow.

It is noteworthy that other studies, not included in this review, are also important for the advancement of software engineering research. During the execution of this review, we came across several papers that provide meaningful contributions to the theory or practice of regression testing research, but exist in an isolated context. The focus of this collection of studies is to find techniques and approaches that are applicable in real software or are close to that - oftentimes, these papers are the result of a longer series of smaller contributions that ultimately culminated in a usable product.

ID	Year	Authors	Title	TCP	TCS	TSR	TSA
S1	2016b	Srikanth et al. [109]	Requirements Based Test Prioritization Using Risk Factors	●	○	○	○
S2	2016	Noor and Hemmati [83]	A similarity-based approach for test case prioritization using historical failure data	●	○	○	○
S3	2016	Schwartz and Do [102]	Cost-effective regression testing through adaptive test prioritization strategies	●	○	○	○
S4	2016	Hirzel and Klaeren [52]	Graph-walk-based selective regression testing of web applications created with Google web toolkit	○	●	○	○
S5	2016	Lu et al. [67]	How does regression test prioritization perform in real-world software evolution?	●	○	○	○
S6	2016	Vöst and Wagner [114]	Trace-based test selection to support continuous integration in the automotive industry	○	●	○	○

S7	2016	Wang et al. [115]	Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search	● ○ ○ ○
S8	2016a	Srikanth et al. [108]	Test Case Prioritization of Build Acceptance Tests for an Enterprise Cloud Application	● ○ ○ ○
S9	2017	Blondeau et al. [12]	Test case selection in industry: an analysis of issues related to static approaches	○ ● ○ ○
S10	2016	Pradhan et al. [90]	Search-Based Cost-Effective Test Case Selection within a Time Budget: An Empirical Study	○ ● ○ ○
S11	2016	Buchgeher et al. [13]	Improving testing in an enterprise SOA with an architecture-based approach	● ● ○ ○
S12	2016b	Tahvili et al. [112]	Dynamic integration test selection based on test case dependencies	● ● ○ ○
S13	2016	Öqvist et al. [93]	Extraction-based regression test selection	○ ● ○ ○
S14	2016	Magalhães et al. [70]	Automatic selection of test cases for regression testing	○ ● ○ ○
S15	2016	Aman et al. [4]	Application of Mahalanobis-Taguchi Method and 0-1 Programming Method to Cost-Effective Regression Testing	● ○ ○ ○
S16	2016	Busjaeger and Xie [14]	Learning for test prioritization: An industrial case study	● ○ ○ ○
S17	2016	Yoshida et al. [122]	FSX: A tool for fine-grained incremental unit test generation for C/C++ Programs	○ ○ ○ ●
S18	2016a	Tahvili et al. [111]	Cost-benefit analysis of using dependency knowledge at integration testing	● ○ ○ ○
S19	2017	Ramler et al. [94]	Tool support for change-based regression testing: An industry experience report	○ ● ○ ○
S20	2016	Strandberg et al. [110]	Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors	● ● ○ ○
S21	2016	Marijan and Liaaen [72]	Effect of time window on the performance of continuous regression testing	● ○ ○ ○
S22	2017	Gotlieb and Marijan [41]	Using global constraints to automate regression testing	○ ○ ● ○
S23	2017	Chi et al. [22]	Multi-Level Random Walk for Software Test Suite Reduction	○ ○ ● ○
S24	2017	Bach et al. [6]	Coverage-Based Reduction of Test Execution Time: Lessons from a Very Large Industrial Project	● ● ○ ○
S25	2017	Spieker et al. [107]	Reinforcement learning for automatic test case prioritization and selection in continuous integration	● ● ○ ○
S26	2017	Vasic et al. [113]	File-Level vs. Module-Level Regression Test Selection for .NET	○ ● ○ ○
S27	2017	Celik et al. [18]	Regression test selection across JVM boundaries	○ ● ○ ○
S28	2018	Ouriques et al. [85]	Test case prioritization techniques for model-based testing: a replicated study	● ○ ○ ○
S29	2017	Kwon and Ko [59]	Cost-effective regression testing using bloom filters in continuous integration development environments	● ● ○ ○
S30	2018	Garousi et al. [40]	Multi-objective regression test selection in practice: An empirical study in the defense software industry	○ ● ○ ○
S31	2018	Shi et al. [104]	Evaluating test-suite reduction in real software evolution	○ ○ ● ○
S32	2018	Haghighatkhah et al. [45]	Test prioritization in continuous integration environments	● ○ ○ ○
S33	2018	Zhang [126]	Hybrid regression test selection	○ ● ○ ○
S34	2018	Miranda et al. [76]	FAST Approaches to Scalable Similarity-Based Test Case Prioritization	● ○ ○ ○
S35	2018	Yilmaz and Tarhan [120]	A case study to compare regression test selection techniques on open-source software projects	○ ● ○ ○
S36	2018	Chen et al. [19]	Optimizing Test Prioritization via Test Distribution Analysis	● ○ ○ ○
S37	2018	Celik et al. [17]	Regression Test Selection for TizenRT	○ ● ○ ○
S38	2018	Zhu et al. [130]	Test re-prioritization in continuous testing environments	● ○ ○ ○
S39	2018	Azizi and Do [5]	Retest: A cost effective test case selection technique for modern software development	○ ● ○ ○
S40	2019	Guo et al. [44]	Decomposing Composite Changes for Code Review and Regression Test Selection in Evolving Software	○ ● ○ ○
S41	2019	Zhong et al. [127]	TestSage: Regression test selection for large-scale Web service testing	○ ● ○ ○
S42	2019	Fu et al. [37]	Resurgence of Regression Test Selection for C++	○ ● ○ ○
S43	2019	Eda and Do [30]	An efficient regression testing approach for PHP Web applications using test selection and reusable constraints	○ ● ● ○

S44	2019	Goyal et al. [42]	Test suite minimization of evolving software systems: A case study
S45	2019	Yu et al. [123]	TERMINATOR: better automated UI test case prioritization
S46	2019	Correia et al. [24]	MOTSD: A multi-objective test selection tool using test suite diagnosability
S47	2019	Machalica et al. [69]	Predictive Test Selection
S48	2019	Najafi et al. [79]	Improving Test Effectiveness Using Test Executions History: An Industrial Experience Report
S49	2019	Leong et al. [62]	Assessing Transition-Based Test Selection Algorithms at Google
S50	2019	Cruciani et al. [25]	Scalable Approaches for Test Suite Reduction
S51	2019	Philip et al. [89]	FastLane: Test Minimization for Rapidly Deployed Large-Scale Online Services
S52	2020	Magalhães et al. [71]	HSP: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study
S53	2019	Wu et al. [116]	A Time Window Based Reinforcement Learning Reward for Test Case Prioritization in Continuous Integration
S54	2019	Land et al. [61]	An Industrial Evaluation of Test Prioritisation Criteria and Metrics
S55	2020	Noemmer and Haas [82]	An Evaluation of Test Suite Minimization Techniques
S56	2020	Lübke [68]	Selecting and Prioritizing Regression Test Suites by Production Usage Risk in Time-Constrained Environments
S57	2019	Yackley et al. [118]	Simultaneous refactoring and regression testing
S58	2019	Shi et al. [105]	Understanding and improving regression test selection in continuous integration
S59	2022	Lima and Vergilio [65]	A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments
S60	2020	Zhou et al. [129]	Beating Random Test Case Prioritization
S61	2020	Peng et al. [88]	Empirically revisiting and enhancing IR-based test-case prioritization
S62	2020	Bertolino et al. [10]	Learning-to-rank vs ranking-to-learn: Strategies for regression testing in continuous integration
S63	2021	Chen and Chen [21]	Multi-objective regression test selection
S64	2021	Rosenbauer et al. [96]	An Artificial Immune System for Black Box Test Case Selection
S65	2022	Bagherzadeh et al. [7]	Reinforcement learning for test case prioritization
S66	2021	Elsner et al. [32]	Empirically evaluating readily available information for regression test optimization in continuous integration
S67	2020	Pan et al. [86]	Dynamic Time Window based Reward for Reinforcement Learning in Continuous Integration Testing
S68	2021	Mehta et al. [73]	Data-driven test selection at scale
S69	2021	Xu et al. [117]	A Requirement-based Regression Test Selection Technique in Behavior-Driven Development
S70	2022	Zhou et al. [128]	Parallel Test Prioritization
S71	2021	Sharif et al. [103]	DeepOrder: Deep Learning for Test Case Prioritization in Continuous Integration Testing
S72	2021	Li et al. [64]	AGA: An Accelerated Greedy Additional Algorithm for Test Case Prioritization
S73	2021	Chen et al. [20]	Context-Aware Regression Test Selection
S74	2022	Zhang et al. [125]	Comparing and Combining Analysis-Based and Learning-Based Regression Test Selection
S75	2022	Abdelkarim and ElAdawi [1]	TCP-Net: Test Case Prioritization using End-to-End Deep Neural Networks
S76	2022	Çıngıl and Sözer [80]	Black-box Test Case Selection by Relating Code Changes with Previously Fixed Defects
S77	2022	Yaraghi et al. [119]	Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts
S78	2022	Omri and Sinz [84]	Learning to Rank for Test Case Prioritization
S79	2022	Greca et al. [43]	Comparing and combining file-based selection and similarity-based prioritization towards regression test orchestration
		Totals	46 41 8 1

Table 2. Selected papers.

3.4 Data Extraction

To collect the data, each of the selected papers was assigned to one author to lead the data extraction, and to another author to review the data afterwards. Thus, each paper was thoroughly reviewed by at least two authors. At the end of this phase, the three authors performed a broad review of the collected information in order to ensure consistency of the results.

During the full-text analysis of the selected papers, we took notes of four groups of properties we wished to extract from each paper. First, we wanted to have the core bibliographical information of the paper. Then, we categorized the papers according to the RT challenge being addressed and contextual factors such as software type and development environment. We also took note of eight properties we considered important regarding IR&A of each proposed technique or case study. Finally, we synthesized the results of the papers by highlighting the types of approaches and metrics used and, if available, the open challenges/future work discussed by the authors. The properties we collected are listed in Table 3.

Some further explanation is needed regarding the "applicability concerns" properties. During the data extraction, it became clear to us that there is some ambiguity regarding industrial motivation; among the non-selected papers, we also saw a great number of them briefly mentioning industry needs in the abstract and introduction, but not forming a connection between those needs and the technique being proposed. So, for our criteria, industrial needs must not only be mentioned, but clearly stated with motivating evidence and/or references, and serve as the actual principle behind the idea of the study.

Regarding the industrial evaluation of results, this property is satisfied when experiments are performed directly on industrial software, usually through collaboration with a technology company. However, there are plenty of papers that have robust experiments performed on notable open-source software, such as those from the Apache and Mozilla foundations. Thus, we collect the following information about the subjects:

- Its openness, which can be industrial proprietary, industrial open-source, fully open-source, or an academic dataset;
- Its testing scale (small up to 500 TCs, medium up to 2,000 TCs, large up to 10,000 TCs, or very large if more than that)[5];
- The language used for writing tests, which can be a programming language, natural language, a domain-specific language or a combination; and
- Its origin (the company that wrote it, or the dataset it is from).

We also checked to see if there is feedback from practitioners in the text of the paper. Relatively few authors include feedback and, often times, it is only a brief passage. Sometimes feedback seems to be implied, but we only considered explicit references to comments from practitioners (in either direct or indirect quotes).

The remaining properties are more straightforward: For experiment subject(s) and industry partner, we merely point out the kind of software (or the specific software, if possible) used for evaluation, as well as any collaboration received from a company. For industrial author(s), we check if the authors of the paper come from an academic, industrial or mixed background. "Available tool" is a URL pointing to an implementation of the technique, if it exists (regardless if it is source code, a plug-in, or a robust replication package). Finally, "put into practice" indicates whether the technique was actually incorporated into the development workflow of a software, to the extent of the information contained in the paper.

[5]It is also important to note that the number of test cases is only one dimension of scale: on S76, for example, evaluation was performed on Smart TV apps with only 38 test cases, but the testing time was over 7 hours.

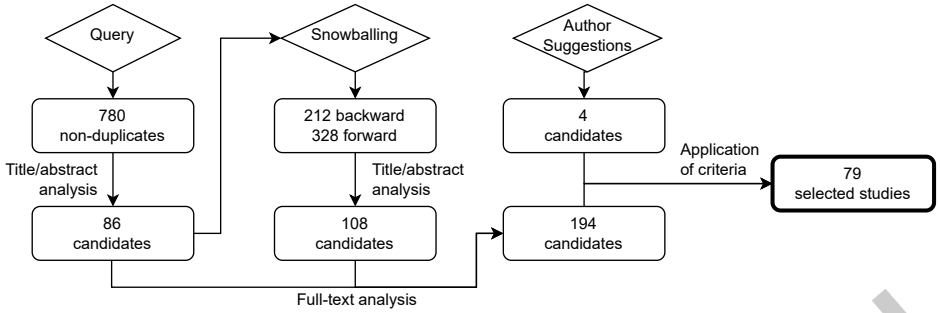


Fig. 1. Diagram of the literature review process.

Bibliographical data	Basic information about the publication.
Date	The date the paper was made available online.
Authors	The list of authors.
Title	The title of the paper.
Abstract	The abstract of the paper.
Venue and Publisher	The conference or journal where it was published and its organization.
DOI	The Digital Object Identifier of the paper.
Categorization	Details regarding the problem addressed by the paper.
RT challenges	Whether the paper covers TCP, TCS, TSR, TSA or a combination.
Context	The type of software targeted by the approach.
Applicability concerns	Properties of the paper related to its IR&A.
Industry motivation	Whether the paper is clearly motivated by an industrially relevant problem.
Industry evaluation	Whether the technique is evaluated in industrial software or sufficiently large-scale open-source projects.
Experiment subject(s)	Which software or kind of software was used for the experimental evaluation of the technique, including the testing scale, the availability and the language in which tests are written.
Industry partner	Which, if any, industrial partner collaborated with the development and/or evaluation of the technique.
Industrial author	Whether one or more of the authors of the paper come from industry.
Practitioner feedback	Whether practitioners were consulted to provide feedback to the results of the paper.
Available tool	Whether the technique introduced in the paper is available to be used, either as a prototype or as a complete tool. If true, we also stored the relevant URLs.
Put into practice	Whether the proposed tool has been adopted into the development process of a certain software.
Findings	Details of the proposed technique and remaining challenges.
Approach	What sort of algorithm and information the technique is using.
Metrics	What criteria are being used for evaluating the techniques.
Open challenges	What the authors list as next steps and unsolved issues related to the problem they addressed.

Table 3. Data extraction form.

Most of the data extracted according to the form can be found in this document under Table 2 (bibliographical data); Table 4 and Figures 2 and 3 (approaches); Table 5 and Figures 4 and 5 (metrics); and Table 6 and Figure 8 (IR&A relevance properties). Additional properties, such as abstracts, DOIs, details on the experiment subjects, and links to supplementary material, can be found in the live repository.

3.5 Questionnaire with Authors

As we collected the data needed to answer the research questions, we realized the need of a perspective beyond what is possible to extract solely from the papers, particularly regarding the ongoing usage of the described techniques. This happens because the information contained in the papers themselves might be out-of-date or

unclear. For example, the authors of S11 mention that their tool, TePSEx, was in use at the time of the publication; however, it is impossible to tell from the paper itself whether the situation has changed since 2016. Conversely, there are also several papers that mention a practical implementation among their future work [S7, S12, S22, S37, S51], but we were not able to find follow-up papers clarifying whether that actually happened.

In order to provide a satisfactory answer to RQ3, we reached out to the authors of the papers via e-mail. Our initial objective was to discover if techniques were ever put into practice and, if so, if they continue to be used to this day. We also realized that the authors could also provide fruitful insight into RQ1 and RQ2, so it ultimately became an important pillar of this work.

We were able to contact authors from most of the papers; in some cases we were not able to locate the author's e-mail address, or the address is no longer valid. The authors were given 12 days to respond and we received replies related to 51 out of the 79 papers — in some cases, one author answered for several papers, in others several authors of the same paper provided answers. The total number of responding authors was 45, although five of them did not provide meaningful answers (i.e. asked us to contact another co-author or said they were not able to answer our questions).

The e-mail sent out to the authors had the following questions:

(1) Is there a functional version of your technique (tool, prototype, source code, etc.) available online? If so, please share with us the URL.
(2) Was there an attempt to implement your technique in industrial or large open-source software? Is the technique currently in use with the software?
(3) If the technique was put into practice, were the metrics used in the paper relevant for the technique's applicability? If not, were there other metrics that proved to be useful?

In addition, we also asked if the authors authorized their answers to be used in this study, if we could link them to their answers, and if they wish to be contacted about updates to this study. All responding authors authorized the use of their answers, but several of them asked not to be linked directly to specific answers due to non-disclosure agreements with industrial partners; therefore, we use the received answers broadly, collecting quantitative and qualitative data from them without specifying which piece of information came from which author. Whenever a direct quote is significant, we transcribe it anonymously; for readability we use an arbitrary ID numbering.

3.6 Survey with Practitioners

In addition to the questions we sent to the authors of reviewed papers, we also prepared a survey destined to practitioners. The objective of this survey was to complement and verify some of the conclusions we draw from the literature, and help align the interests of academia and industry.

The survey was disseminated using a convenience sample, including contacts we personally know in industry and people who participated in software testing centric events. We considered also putting the survey in public online forums centered on software testing/engineering, but ultimately decided against that for fear of low-quality responses and data pollution.

We received 23 responses from practitioners in six different countries (Brazil, Italy, Finland, Hungary, Portugal and Sweden). Obviously this survey covers an extremely small part of software testing practice, but it is possible to trace some common elements pointed out by the respondent that corroborate some of the findings and conclusions we had extracted from the literature. When relevant, these responses are used in Section 9.

Due to space concerns, we cannot include the full questionnaire here, but it is available online, along with the anonymized responses we received. The main points covered in it were:

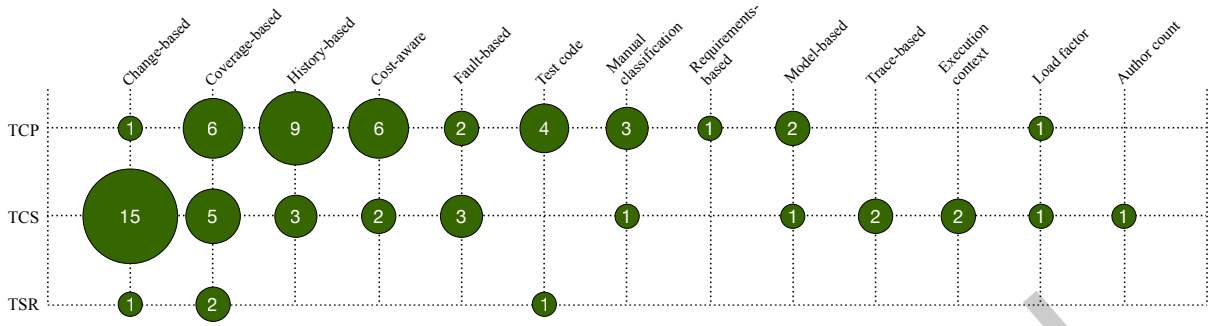


Fig. 2. Distribution of information approaches.

(1) What are the most common pain points when it comes to regression testing?
(2) Do you know of, or have you ever used, a regression testing tool originating in academia? If so, how was the experience of using it?
(3) Do you stay informed on current advances in software engineering research? Are there attempts of collaboration between your company and academia?

We also asked their company, country and role, which we used to assess the diversity of respondents. This will not be published for privacy reasons.

3.7 Replicability

To allow replicability of our review and clearly describe the thought process behind the choice of included studies, we make a replication package available online[6]. This package includes the original search queries, the list of papers that we included or excluded via the criteria, the full contents of the data extraction form, and the data used to generate the figures. It also includes the full version of the e-mail template sent to the authors and the full questionnaire sent to practitioners.

4 RQ1: COMMON APPROACHES AND METRICS IN RT RESEARCH

A summary of the main approaches used to tackle RT challenges is presented in Table 4. We observe that the approaches adopted by a technique may serve to two different purposes: one is regarding the source from where the information used as input for a technique is collected[7]; a second purpose refers to the actual algorithm used to address the problem to solve. Correspondingly, the main approaches used to tackle RT challenges are presented in Figure 2 and in Figure 3, respectively. Regarding information, change-based, coverage-based, history-based and cost-aware approaches are the most common; while machine learning-based, search-based, similarity-based and graph-based are the popular algorithmic approaches.

The main metrics reported in the literature are shown in Table 5, Figure 4 and Figure 5, grouped according to their main goal[8]. The reported metrics primarily focus on effectiveness (how good a solution is at accomplishing its task) or efficiency (the time and cost of using the solution), but two metrics were identified that are neither — namely, applicability/generality and diagnosability.

APFD is the most widely accepted metric for assessing TCP approaches. Because TCS and TSR both have the goal of running fewer tests than an original test suite, their metrics are mostly shared: testing time, selection

[6]Available at: https://doi.org/10.5281/zenodo.7514251.
[7]This is also referred to as criteria in the literature [66].
[8]In each figure, we omit TSA due to space concerns, as only one paper [S17] covers it.

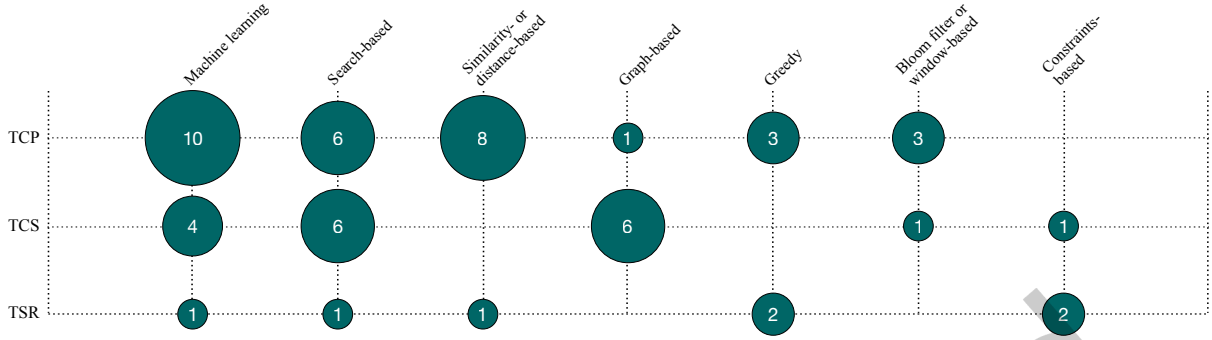


Fig. 3. Distribution of algorithm approaches.

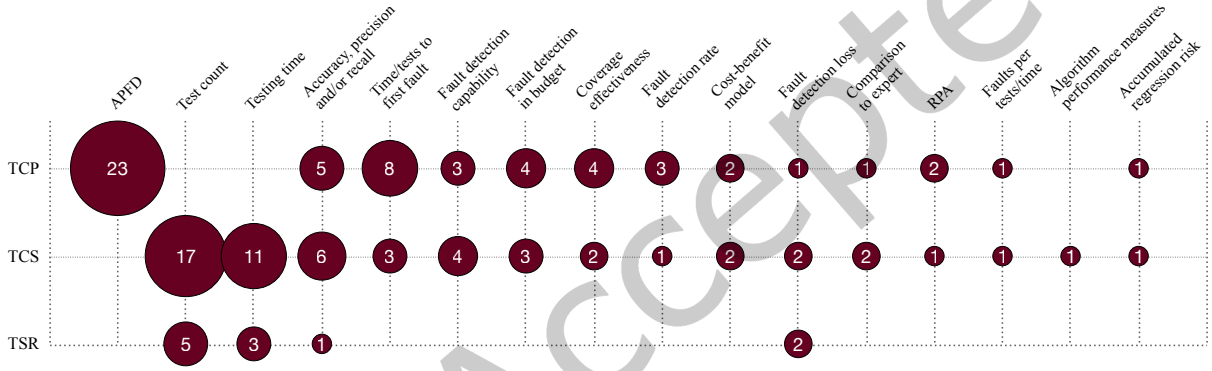


Fig. 4. Distribution of effectiveness metrics.

count and fault detection ability are the most common ones. The set of accuracy/precision/recall appears to be the effectiveness metric that covers the most situations. For efficiency, the execution time of a technique is both widely used and is useful for any kind of solution.

In our questionnaire to the authors, we included a question focused on the choice of metrics. We asked authors who had successful or attempted attempts of implementing their technique whether the metrics described in the paper proved to be relevant in practice, or if additional measures were needed. We received 27 meaningful responses to that question, out of which 24 were satisfied with the chosen metrics. We quote some of the answers received: "*The metrics directly influenced decisions of the industrial partner*" (respondent author #16); Respondent author #8 stated that "*[the] metrics were at the heart of the approach*" and that the provided metrics were "*always perceived as necessity by developers to support them in their work*"; "*The technique was put in practice for subsequent release and the metrics were useful and effective*" (author #17); Respondent author #23 answered that "*the metrics presented in the paper were critical for adoption and to measure ongoing improvements*"; "*they [the metrics] were relevant - they were also collected in the same environment in which the technique ended up being used*" (author #19).

Out of the three divergent responses, one suggested that the metrics were not a problem, but the dataset they used for the experiments was too small to provide meaningful evidence (author #42). Curiously, the remaining two complement each other. Author #45 said that they proposed a new metric, which is believed to be relevant

Information	TCP	TCS	TSR	TSA	Description
History-based	S8, S15, S16, S20, S21, S29, S32, S45, S48	S29, S47, S48			Uses information from previous testing cycles to decide about test case relevance.
Change-based	S20	S13, S19, S26, S27, S33, S35, S37, S40, S42, S43, S58, S73, S74, S76, S79	S43		Uses changes between versions to identify the relevant test cases.
Coverage-based	S16, S24, S28, S45, S52, S56	S19, S24, S41, S52, S56	S31, S55	S17	Uses structural coverage information, whereby coverage can be of statement, method, class, file, etc.
Cost-aware	S5, S7, S12, S18, S45, S70	S12, S63			Utilizes test case cost or time information to assess test relevance.
Requirements-based	S1		S44		Relate tests with project-sensitive information, such as requirements and risk assessments.
Manual classification	S12, S18, S45	S12			Requires at least some information that must be manually inputted by an expert.
Model-based	S11, S28	S11			Informs the test technique using behavioural or architectural models.
Trace-based		S6, S41			Provides inputs and keeps track of the effects of those inputs throughout the program.
Fault-based	S21, S38	S49, S63, S76			Utilizes information related to fault detection or failure beaviour.
Test code	S2, S34, S61, S79		S50		Uses the plain text source code of the test cases.
Load factor	S11	S11			Indicates what parts of the SUT are most used by different services and components.
Author count		S49			Number of authors associated with a certain part of the SUT.
Execution context		S27, S73			Considers environment data such as libraries, APIs, databases, operating system, etc.
Algorithm	**TCP**	**TCS**	**TSR**	**TSA**	**Description**
Similarity / distance-based	S2, S15, S16, S28, S32, S34, S60, S79		S50		Assesses test cases based on their similarity, with the objective of diversifying the suite.
Search-based	S5, S7, S46, S52, S61, S70	S10, S14, S30, S46, S52, S64	S23		Utilizes search-based algorithms, such as genetic or nature-inspired ones
Machine learning-based	S16, S25, S53, S59, S62, S65, S66, S75, S77, S78	S47, S66, S68, S74	S51		Trains a ML model using historical or other data. Includes supervised, unsupervised and reinforcement learning methods.
Graph-based	S28	S4, S19, S35, S37, S39, S62			Creates a graph representation of the SUT and utilizes graph theory algorithms to solve problems.
Greedy	S5, S70, S72		S24, S55		Utilizes greedy algorithms and heuristics (usually based on coverage or similarity information).
Constraints-based		S43	S22, S43		Utilizes constraint programming paradigm.
Bloom filter, window-based	S3, S29, S67	S29			Utilizes Bloom filter data structures and time windows to filter out tests that fail only once.

Table 4. Information- and Algorithm-based Approaches

but has not been experimented in practice yet; while author #25 claimed their own choice of metrics was not relevant to applicability, and is considering using the same metric proposed by #45.

After analyzing all the answers to this question, two very interesting things emerged: 1) One author (#14) reflected that although the metrics used were relevant at the time, looking back in retrospect other relevant metrics should have been used — "*Now, 7 years later, we have realized that some metrics were not included that should have been included*". The author was referring to the use of a metric for test case diversity as this could have helped them to tune the approach to avoid putting together many test cases targeting the same functionalities. This reinforces the importance of following up the adoption of a proposed approach in its application environment: even if we strive to anticipate all the possible uses of a proposed approach, observing its adoption in a real industrial context may reveal details and needs that were not captured while the approach was being conceived. 2) Two respondent authors (#23 and #36) reported that their approaches were evaluated with some additional

Effectiveness	TCP	TCS	TSR	TSA	Description
Selection/reduction count/percentage		S4, S6, S9, S24, S26, S27, S33, S35, S37, S39, S42, S43, S47, S58, S73, S74, S76	S22, S23, S43, S44, S55		Absolute or relative size of the resulting test suite compared to the original.
Average Percentage of Faults Detected (APFD)	S1, S5, S8, S16, S21, S25, S28, S32, S34, S36, S45, S53, S59, S61, S65, S67, S70, S72, S75, S77, S78, S79				A measure of how quickly a test suite detects faults, on average. Includes many variations, such as APFDc and NAPFD.
Testing time		S12, S19, S27, S30, S35, S37, S41, S58, S68, S74, S76	S44, S51, S55		Time required to execute the prioritized/selected/reduced test suite as opposed to the original suite.
Accuracy/precision/recall	S16, S29, S67, S75, S78	S9, S14, S29, S40, S47, S69	S51		Measures of correctness and completeness of the resulting test suite (e.g., count of false positives and false negatives).
Fault Detection Capability	S3, S7, S21	S29, S64, S73, S76			Number or proportion of faults detected by the resulting suite compared to the original.
Fault Detection Rate (FDR)	S15, S20, S45	S39			Time to detect faults compared to the optimal RT suite.
Coverage Effectiveness (CE)	S2, S45, S52, S56	S52, S56		S17	Measure of the tradeoff between cost of the test suite and structural coverage of the SUT.
Time/tests To First Failure	S2, S36, S38, S59, S60, S67, S70, S79	S9, S64, S79			Number of tests or amount of time needed to reach the first failure.
Fault detection within a budget	S7, S24, S59, S79	S10, S24, S79			Faults still detected when restricting the testing time budget.
Cost-benefit model	S3, S18	S30, S68			Mathematical models considering costs and benefits of applying a technique throughout development.
Fault Detection Loss	S48	S48, S63	S31, S50		Number or proportion of faults undetected by the selected/reduced test suite compared to the original.
Comparison to expert	S11	S11, S14			Compares the output of the tool with a list of tests selected by the project architect.
Faults per tests/time	S29	S29			Number of faults deteted per number of tests or testing time.
Number of tests added				S17	Number of tests added to the test suite.
Algorithm performance measures		S10			Fitness value or hypervolume metrics applied to search-based algorithms
Accumulated regression risk	S56	S56			How much of the "regression risk" is covered by the tests.
Rank Percentile Average (RPA)	S62, S65	S62			Comparison between the predicted ranking and the actual ranking (from the dataset).
Efficiency	**TCP**	**TCS**	**TSR**	**TSA**	**Description**
Execution time	S7, S32, S34, S48, S53, S59, S70, S72, S79	S4, S26, S27, S41, S48, S69, S74, S79	S22, S23, S50	S17	Time required to run the tool (e.g., selection time, prioritization time, etc).
Total/End-to-end time	S7, S34, S62, S79	S13, S26, S33, S37, S42, S62, S74, S79			End-to-end time, combining measuring time, execution time and testing time. Due to this, it is a measure of both efficiency and effectiveness.
Memory usage	S7	S4			Measures the amount of memory used by the tool.
Scalability	S34, S77		S50		How well the tool performs on subjects of different sizes.
Measuring time/cost	S66	S66			Measure of how costly is the information needed by the technique (e.g. compiling tests, collecting coverage, training a model).
Other	**TCP**	**TCS**	**TSR**	**TSA**	**Description**
Applicability/Generality	S60	S69			The variety of SUTs upon which the tool can be applied.
Diagnosability	S46	S46			Cost of diagnosing a fault upon detection.

Table 5. Effectiveness, Efficiency and Other Metrics

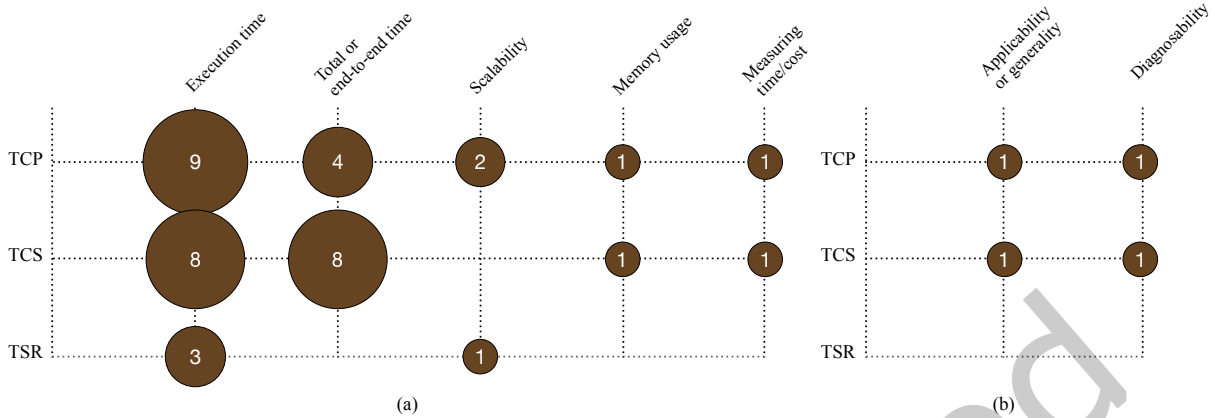


Fig. 5. Distribution of (a) efficiency and (b) other metrics.

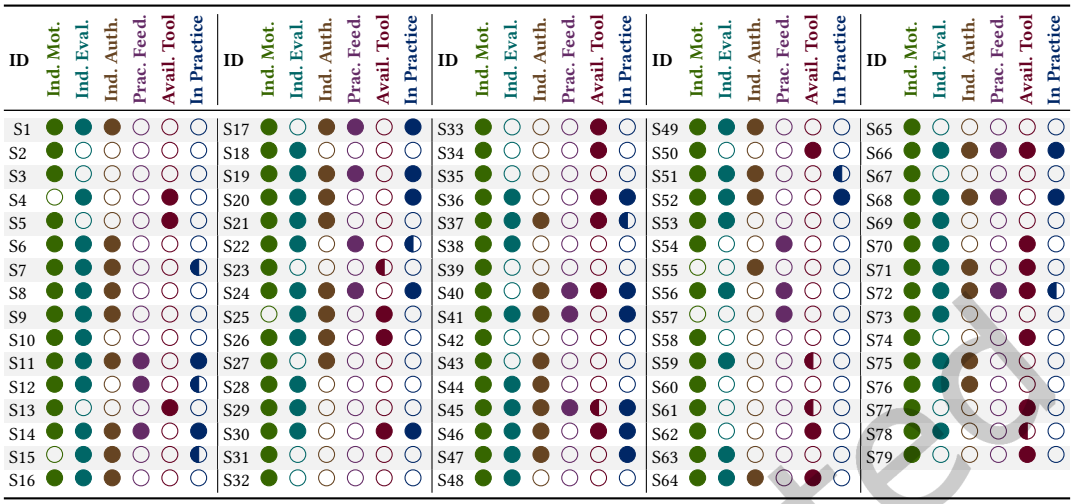
metrics relevant to industry — "*the company has also developed their own metrics*" (respondent author #36) — that were not reported in their papers. The answers do not make it clear if the metrics were omitted because the measurements were not available at the time the paper was published or if they were omitted on purpose (e.g., because they could reveal sensitive company data).

Summary of RQ1. The data reported in the figures show what are the most common approaches and metrics according to the objective of the RT techniques. For example, we see that TCP often relies on history-based and similarity-based approaches and uses APFD for evaluation, while TCS is usually change-based with a focus on the number of selected tests. We can also see that some overlap occurs and there are authors who choose unconventional but potentially promising combinations of techniques and metrics. From the author responses we received, it appears that many authors are satisfied with their selection of metrics but a few indicate that more were discovered in the process of implementing the tool with their industrial partner.

5 RQ2: APPLICABILITY CONCERNS IN REGRESSION TESTING RESEARCH

To answer this research question, we look carefully at the applicability concerns extracted according to Table 3. The full mapping of the papers with the properties they satisfy is available in Table 6. It is worth observing that our conclusions here, as well as in the next section, are only relative to the set of primary studies that we retrieved; we cannot exclude the possibility that works that we did not select could eventually find application in practice. For instance, a paper with no obvious practical motivation could be the theoretical foundation for a tool later adopted by practitioners.

Most of the selected papers satisfy the properties of having a clear industrial motivation: out of the 79 papers, only five [S4, S15, S25, S57, S58] did not have a clear IR&A motivation. Regarding evaluation, 50 of the papers contained experiments on industrial (or industrial-scale) software. In other words, it is quite clear that IR&A is frequently a concern that motivates researchers to develop novel RT techniques. While providing adequate experimentation and evaluation to these techniques can be a tough challenge, it is one that researchers are indeed attempting to address.



ID	Ind. Mot.	Ind. Eval.	Ind. Auth.	Prac. Feed.	Avail. Tool	In Practice
S1	●	●	●	○	○	○
S2	●	○	○	○	○	○
S3	●	○	○	○	○	○
S4	○	●	○	○	●	○
S5	●	○	○	○	●	○
S6	●	●	●	○	○	○
S7	●	●	●	○	○	◑
S8	●	●	●	○	○	○
S9	●	●	●	○	○	○
S10	●	●	○	○	○	○
S11	●	●	●	●	○	●
S12	●	●	○	●	○	◑
S13	●	○	○	○	●	○
S14	●	●	●	●	○	●
S15	○	●	●	○	○	◑
S16	●	●	●	○	○	○
S17	●	○	●	●	○	●
S18	●	●	○	○	○	○
S19	●	●	●	●	○	●
S20	●	●	●	○	○	●
S21	●	●	●	○	○	○
S22	●	●	○	●	○	◑
S23	●	○	○	○	◑	○
S24	●	●	●	●	○	●
S25	○	●	●	○	●	○
S26	●	●	●	○	●	○
S27	●	○	●	○	○	○
S28	●	●	○	○	○	○
S29	●	●	○	○	○	○
S30	●	●	○	○	●	●
S31	●	○	○	○	○	○
S32	●	○	○	○	○	○
S33	●	○	○	○	●	○
S34	●	○	○	○	●	○
S35	●	○	○	○	○	●
S36	●	●	○	○	●	●
S37	●	●	●	○	●	◑
S38	●	●	○	○	○	○
S39	●	○	○	○	○	○
S40	●	○	●	●	●	●
S41	●	●	●	●	○	●
S42	●	○	○	○	○	○
S43	●	○	●	○	○	○
S44	●	●	●	○	○	○
S45	●	●	●	●	◑	●
S46	●	●	●	○	●	●
S47	●	●	●	○	○	●
S48	●	●	○	○	○	○
S49	●	●	●	○	○	○
S50	●	○	○	○	●	○
S51	●	●	●	○	○	◑
S52	●	●	●	○	○	●
S53	●	●	○	○	○	○
S54	●	○	○	●	○	○
S55	○	○	●	○	○	○
S56	●	●	○	●	○	○
S57	○	○	○	●	○	○
S58	●	○	○	○	○	○
S59	●	●	○	○	◑	○
S60	●	○	○	○	○	○
S61	●	○	○	○	◑	○
S62	●	○	○	○	●	○
S63	●	●	○	○	○	○
S64	●	●	●	○	●	○
S65	●	○	○	○	○	○
S66	●	●	●	●	●	●
S67	●	○	○	○	○	○
S68	●	●	●	●	○	●
S69	●	●	○	○	○	○
S70	●	●	○	○	●	○
S71	●	●	●	○	●	○
S72	●	●	●	●	●	◑
S73	●	●	○	○	○	○
S74	●	○	○	○	●	○
S75	●	●	●	○	○	○
S76	●	●	●	○	○	○
S77	●	○	○	○	●	○
S78	●	●	○	○	◑	○
S79	●	○	○	○	●	○

Ind. Mot.: Industrial Motivation. **Ind. Eval.**: Industrial Evaluation. **Ind. Auth.**: Industrial Author(s). **Prac. Feed.**: Practitioner Feedback. **Avail. Tool**: Available Tool. **In Practice**: Put into Practice. A half-filled circle indicates a partially satisfied property. For example, a paper that provides its dataset but not its source code, or one that has some indication of having been implemented without explicitly stating so.

Table 6. Relevance properties found in the papers.

Out of the 74 papers with relevant evaluation, 44 perform experiments with the direct collaboration of an interested partner — in most cases a corporation, in one case a government department [S30], indicating that such collaborations can play an important role in improving the relevance of experiments. Curiously, there are also four papers that have industrial collaborations, but the experiments are not performed with software from that partner [S17, S27, S40, S55]. Finally, there is one paper with an industrial partner but the objective of the work was not to develop a tool, so there are no experiments [S54].

In our retrieved literature, the industrial background of the authors is significant in a few ways. The papers with primarily industrial authors are the most likely ones to be relevant in practice, because these are generally designed with the application on a specific software product in mind; these papers usually provide insight into the testing workflow at large companies and share the lessons learned from applying a certain technique to a specific scenario. Examples include S47 with Facebook; S49 with Google and S51 with Microsoft. There are also some cases of companies whose main product is not software, but software is an important part of their products (e.g., transportation manufacturers, as S6 with BMW).

Papers with a mix of industrial and academic authors also represent good progress in enhancing industry-academia collaborations, such as the collaborations between University of Texas at Austin and Microsoft [S26, S37] or between the Federal University of Pernambuco and Motorola [S14, S52].

Finally, we want to highlight the papers that have tools available online. This is important for replicability and ease of access, but is still lacking in many publications. To facilitate comparisons by other researchers and simplify experimentation by software developers, it is fundamental that a version of the technique exists, either in binary or source code format. Only 22 of the surveyed papers made their tool available in some form (usually source code repository), making it improbable that any of the other tools were used by practitioners without direct contact with their developers. Notably, there appears to be a change in this trend: between 2016 and 2020, only 14 papers had any sort of replication package or tool available. In 2021 and 2022 (up until July), we found 8 papers satisfying this criteria. The likely explanation to this is that noteworthy Software Engineering conferences have given more value to easily-replicable research in recent years and this has caused authors to make it a priority.

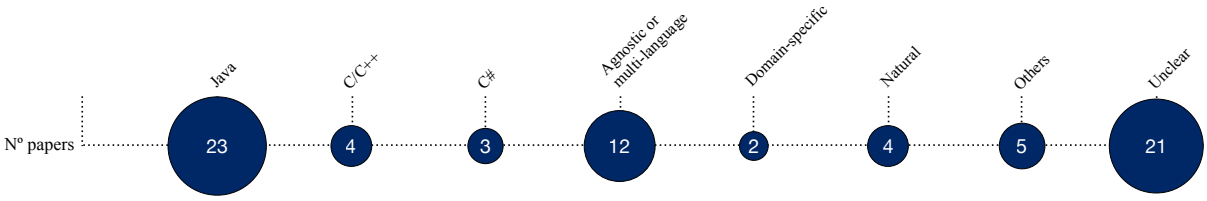


Fig. 6. Distribution of the targeted programming languages.

However, there are some cases where the code is made available with little to no documentation or explanation of how it works; on the bright side, there are also examples that stood out for having clear and detailed steps on how to use the code and replicate the experiments. Among the e-mail responses from the authors, we received the source code repository URL for four additional papers, confirming that at least 26 papers have material available online — whenever possible, the relevant URLs can be found in our live repository (Section 8).

Out of the investigated URLs, only S4 and S33 provide clear usage instructions for arbitrary software projects; they are available as plug-ins for the Eclipse IDE and the Maven build system, respectively. S40 also mentions the tool is available as an Eclipse plug-in, but we were unable to find a URL pointing to it. The remaining papers provide their source code primarily for study replication, not necessarily intended for actual usage by developers, meaning that the tool is likely not sufficiently robust for practical usage beyond experiments. It also happens frequently that tools developed in conjunction with an industrial partner end up becoming proprietary software and cannot be easily distributed (e.g., S14, S16). Authors of 14 papers said in their responses that the code or the tool could not be shared, since the resulting software is completely or partially proprietary or confidential.

An issue we identified is regarding the programming languages of the SUTs targeted by the experiments. Figure 6 shows that there is a heavy bias towards Java, with 23 papers targeting software written in that language. On most of the papers focused on a specific language, it is not clear if the same approach would be easy to adapt and would produce equivalent results on software developed using other widely used languages. However, 12 papers target systems written in multiple languages, or explicitly state that the approach is language-agnostic, which highly increases its applicability. Unfortunately, it was not possible to identify the target language of 21 papers; this creates a substantial challenge for both the replicability of the experiments and applicability of the technique.

Summary of RQ2. Our survey shows that a large number of papers exhibit IR&A concerns in their motivations, and a smaller albeit still significant amount contains experiments at relevant scale. Most of the times, the techniques that are implemented into a software workflow are also papers that have authors from an industrial background. Unfortunately, few authors share their tools in a well-documented, open-source fashion, which hampers both future researchers, who wish to compare their solutions against the state-of-the-art, and practitioners, who might want to see how existing RT tools can help their software.

6 RQ3: EVIDENCES OF REAL-WORLD APPLICATION OF REGRESSION TESTING TECHNIQUES

Our study is motivated by the concern that there is potentially valuable technology being proposed in academia that does not always make its way into usage in industry. The difference between the state-of-the-art techniques proposed in academia and the ones actually used in real-world software is what we call the *academia-industry technology transfer gap*. Expressing concerns over IR&A of RT techniques is an important step towards awareness of the

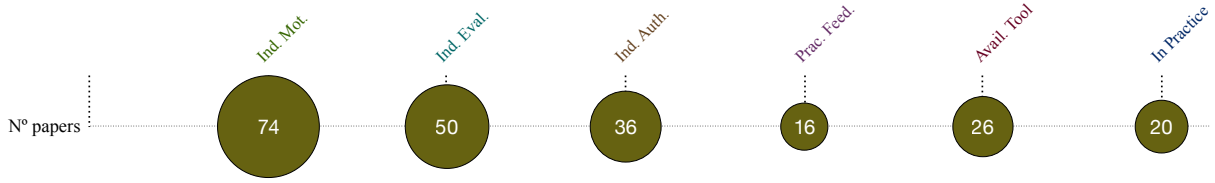


Fig. 7. Quantitative analysis of the satisfied criteria.

gap, although not sufficient *per se* to solve the problem of actually putting these techniques into practice. The focus of this section is to discover if and how much evidence exists of techniques developed by the research community being adopted by real-world software development. As previously stated, there might be studies that have been put into practice, but escaped our review because they were not explicitly motivated by IR&A; we hope that, in the future, our live repository solution will eventually find them and potentially widen the conclusions described here.

Table 6 contains only data extracted from the papers themselves; since the author responses are anonymous, we cannot map them directly to the table. Thus, Figure 7 displays the total number of papers that satisfy each of our applicability criteria, including updates from the author responses. In other words, we consider the author response if it updates the information retrieved from the paper; otherwise the data extracted from the paper remains.

Regarding the adoption of the proposed approaches, Table 6 shows that 16 out of the 79 selected papers explicitly state that the proposal is applied with a partner, or suggest that implementation was ongoing at the time of publication, out of which six are confirmed to still be in use by their authors, while four say it fell out of use (the remaining six did not respond, so we assume no change). Eight other authors claim their approach was implemented after publication, so the count in Figure 7 is 20 (16-4+8).

We can observe that having a practitioner as a co-author helps to provide a direct line from the founding theory of the technique to its application in practice: indeed, 14 of these 20 papers have at least one author from industry. This is not surprising, because such collaborations often originate directly from a need expressed by the practitioners.

However, we also see that only 8 out of those 20 papers featured feedback from the practitioners who actually used the developed tool. That is, although the tool was incorporated into the production workflow, in many cases an assessment of long-term benefits and acceptance by its users is either not done or not reported. Ultimately, the authors were our best chance of understanding the story behind each tool, revealing whether it is still being used by a partner and the reasons it might have fallen out of use.

From the respondent authors, we received six confirmations that the tool continues to be in use by their industrial partner in some form, e.g. "*The tool was implemented at a company [...] and it is still in use at the company [with significant changes].*" from respondent author #14. Authors of another two papers stated that the technique is undergoing an implementation process at the time of the response. Author #37 claims that their work on a newer paper is seeing adoption by an industrial partner; however, at the time of writing, that paper remains in pre-print and cannot be formally included in this review.

Interestingly, eight authors say that the tool was successfully incorporated into an industrial partner's development cycle after the publication of the paper: "*the technique has been adapted and embedded into a random data selection tool by the [company]'s testing team, for purposes including but not limited to regression testing.*" (author #36); "*the [technique] has been in use at [company since roughly the date of publication. [It] is used to run relevant test cases for every code review in [company]'s main code repository.*" (author #23). However, the details are not always known to them: "*We were told it was put into practice but we were not given any information, due to confidentiality rules.*" (author #44).

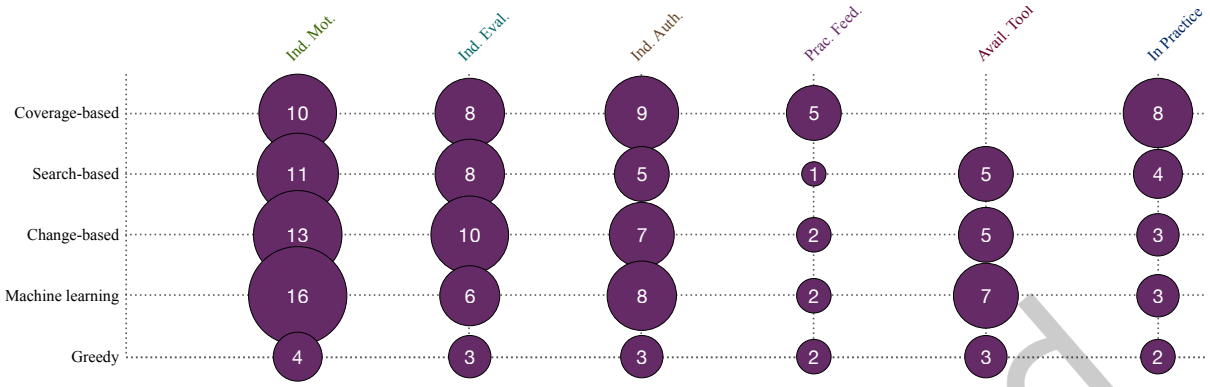


Fig. 8. Mapping of approaches and techniques that have seen practical application on at least 2 papers.

To the extent of the authors' knowledge, 12 papers were never put in practice, although some say there was a discussion to do so at some point. From author #35: "*We discussed the possibility of conducting a research visit at one of the corporation branches to experiment with the technique* in vivo*, but in the end it did not go through.*"

Authors of ten papers (out of which four were tagged as implemented in Table 6) said that the tool saw usage but fell out of use after a few years; an additional three claimed some sort of attempt, but the current status is unknown. What this means is that, even if a technique is incorporated into a software, a lot of work must still be done to ensure that the approach remains viable in a longer term. Some challenges mentioned by these authors include:

The tool became outdated and it was not updated to remain relevant. "*It was implemented in an industrial setting, but this work is several years old and has to be evolved to stay relevant for business.*" (author #20). This can be either due to a technical issue, e.g. the tool was designed for an older version of a programming language or platform and would require some effort to be updated and be used on newer software, or because the tool does not consider newer requirements of its subject software.

The authors noticed that adapting an academic prototype into an industry-strength tool required more time and budget than the project permitted. "*There is a gap between developing a research prototype and an industrial-strength tool. Evolving research prototypes towards industry-ready tools was beyond the project budget.*"(author #8) It can happen that a technique seems promising in initial experiments, but an enormous amount of work would be needed to actually incorporate it into a workflow. The technique might require data that is not currently being collected, or use some manual process for the evaluation that would need to be automated. The tool must also be verified for correctness and robustness before practical usage.

Authors lost contact with their partners and no longer follow development of the tool. "*[The tool] was supported by [our partner]. We have no input if the tool has been used.*" (author #26) There are cases where the partnership does not continue after the publication of the paper or some other condition occurs. The industrial partner is likely free to continue using the developed tool, but the authors from the academic side are no longer part of its evolution and do not receive updates and feedback regarding the subsequent challenges and achievements.

The cost-benefit ratio was off. "*We tried to use it within [our partner]. It seemed to work fine but the cost associated with the 1% bugs that were missed is too high.*" (author #43) Even if a TCS technique detects 99% of bugs by running a very small set of tests, practitioners will be skeptical of using as a replacement for TestAll strategy. After all, although testing is a costly procedure, it is still much cheaper to detect an error during testing than after the software has been shipped to customers.

Figure 8 shows the relationship between the applicability criteria and the approaches that have seen real-world usage. The figure shows approaches with at least two papers put into practice[9]. Unsurprisingly, the most common information-type and algorithm-type approaches are the ones that see the most real-world usage. Coverage-based approaches dominate the implementations of techniques, despite previous concerns regarding the cost of measuring coverage [50]; although time-consuming, coverage measurements are easy to obtain in most programming languages. Conversely, there are 16 papers proposing machine learning approaches, but only three were implemented, likely because machine learning models are only as good as the data they are fed; often, obtaining data of enough volume and quality is more difficult than implementing the method itself.

From the practitioners' point of view, one possible source of information is grey literature — that is, material produced by experts and published without peer-review. However, this data is decentralized and unstructured, making it difficult to locate useful information. We did find one example: Netflix has a post on their blog [57] describing a system they developed inspired by S25. This indicates that grey literature might be worthy of investigation, but such an effort would fall beyond the scope of the current study.

To provide some insight into the state of practice, we surveyed 23 practitioners who are involved with software development and/or testing at their workplace. 60% of respondents claim they do not know about RT tools that originated from research, which corroborates the well-known lack of communication. 35% say they use or have used a tool to aid RT; however most of these claim the tool was developed specifically for their needs, so it is not clear that their origins can be traced back to Software Engineering research.

Summary of RQ3. From the papers and the responses we received, we have evidence that 20 papers propose techniques that are still being used in practice. It is a relatively small number, but it shows that RT research can have concrete positive impact on real-world software development. Unfortunately, many of the techniques that are implemented fall out of use after some time, as an ongoing effort is needed to motivate their usage and keep the tool relevant and updated. There is a hint of evidence stemming from grey literature, although practitioners themselves, when surveyed, mostly claim to be unaware of RT techniques originated in academia.

7 THREATS TO VALIDITY

In this section, we present the potential threats to the validity of our results.

Construct validity. Despite our efforts to comprehensively find all primary studies that meet our selection criteria, we might have missed some. To mitigate this threat, we performed a systematic search over five broad digital libraries and complemented the search with a snowballing cycle and a check with authors of all found studies, who in fact suggested a few additional entries.

As usual for this kind of study, our selection of papers was performed through queries, followed by manual filtering. To diminish potential bias of the latter step, the filtering process was systematically reviewed and agreed upon among all the three authors.

Internal validity. The internal validity of this study is strongly dependent on the three research questions that guided all our analysis as well as the data extraction form we built. We took great care in ensuring that they properly reflect our objectives, although it is unavoidable that, by formulating different questions or using other data extraction forms, we could have obtained other results. We might also have overlooked or misinterpreted some important information or arguments in the primary studies, beyond our best efforts and accuracy in the full reading of all selected papers. To mitigate such threats we provide all extracted data in traceable format,

[9]Constraint-based, graph-based, similarity-based, trace-based, manual classification, cost-aware and history-based approaches have one paper each implemented in practice.

highlighting the main points we extracted from each primary study. Furthermore, the responses we received directly from authors often provide additional context that reduce the risk of misinterpretation. That said, we cannot make the full responses available due to non-disclosure requests from some authors.

Conclusion validity. The conclusions we drew in terms of the information we summarize from the primary studies, the detected challenges we discuss in the above section and the recommendations we formulate in the conclusions might have been influenced by our background, and other authors might have reached different conclusions. Such potential bias is unavoidable in this type of study, however we tried to mitigate it by aiming at full consensus of all authors behind each conclusion. Furthermore, by documenting in detail the data extraction process, we ensure a fully transparent study that can be verified and replicated. The survey sent to practitioners helps to validate our conclusions. Although the sample of 23 responses is very small, it shows a degree of alignment among people working in six different countries. A convenience sample was used to distribute the survey; thus, the practitioners we reached are more likely to have some contact with ongoing research. To avoid excessive bias in that direction, we did not contact members of industry who are known to regularly publish in Software Engineering events.

External validity. We do not make any claim of validity of our conclusions beyond the 79 papers analyzed. As more primary studies are published, they should be read and analyzed on their own, and our conclusions should be revised accordingly. In consideration of this threat, in the aim of ensuring validity even in future, we are committed to keep the live repository up-to-date, taking into account the community inputs. Moreover, we believe that the framework we developed consisting of the three research questions, the data extraction form and the structured tables for summarizing the approaches and the metrics could be still applicable also by other external authors.

8 LIVE REPOSITORY

Literature reviews provide important information to researchers starting out in a field or practitioners who are curious to know the latest innovations, but do not have time to fully explore journals and conferences. However, it is inevitable that a literature review such as this becomes outdated after some time, as new research comes out that cannot be included in the published paper. This of course limits the long-term value of the work, since the text will no longer reflect the ongoing research in the field.

In order to aggregate long-term value to this work, we have made the list of papers and the information extracted from them available as an online live repository[10]. The papers in Table 2 serve as the starting point for a list that will continue to grow year over year. We hope this website will serve as reference to anyone who is interested in practical applications of regression testing techniques in the coming years.

The main challenge is how to keep this repository alive in the long term. It is unfeasible for us to add a relevant paper to the repository as soon as it is published, so our plan is to update the list in a yearly basis, re-running the query and screening steps detailed in Section 3. That way, we can at least assure the most recent paper included is no more than one year old. We are also looking into the possibility of getting automatic notifications when a paper that satisfies certain criteria is published in an online library. For now, this work is done by the original authors of the literature review; according to future necessities, we will appoint other researchers or graduate students to help with the process. In addition, we also encourage authors to submit their own work by filling a form linked on the website.

The repository also contains a separate section for relevant literature reviews. This is initially populated by the reviews mentioned in Section 2 and, upon publication, this very document. With this we aim to provide a starting point for new researchers and a place to gather the overarching themes of the field.

It can also happen that, over the years, the definitions we selected for including a paper in the repository must be adjusted. Whenever an author submits a paper, we will use the opportunity to consider whether or not the paper itself is a good fit for the repository, but also if there are new trends that our existing selection process does not account for. There will likely be a point in the future when the industry/academia landscape has shifted and

[10]Available at: https://renangreca.github.io/literature-repository/.

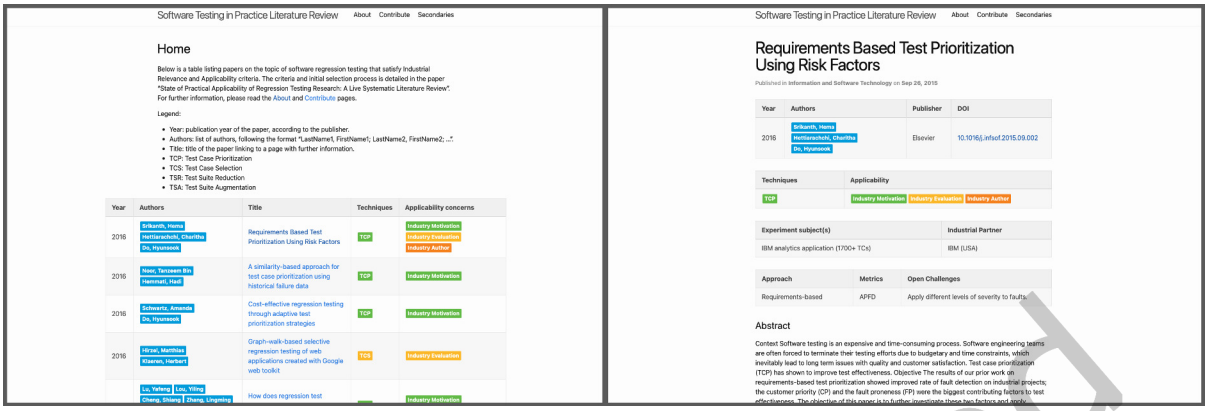


Fig. 9. Screenshots from the live repository. From left to right: 1) the main page listing the included papers; and 2) a single paper's page (S1 used as example).

ID	Title	L	A	P	ID	Title	L	A	P
CH1	Alignment of motivations			●	CH5	Converting research into usable tool		●	●
CH2	Realistic experimentation	●			CH6	Absence of TSR/TSA	●		●
CH3	Scalability	●			CH7	Clarity of target	●		
CH4	Relevance of metrics	●	●		CH8	Communication		●	●

Source(s): **L**: Literature; **A**: Author questionnaire; **P**: Practitioner survey.

Table 7. Summary of main challenges identified by this study.

this study will no longer be needed. When that happens, we will discuss the possibility of freezing the repository and stopping further expansions.

Aside from newer papers, it is always possible that we have missed some relevant papers for a variety of reasons, so the live repository is another way of mitigating that risk. It is impossible to provide a complete and definitive overview of any field, but we believe that a live repository is the closest approximation that can be expected.

9 CHALLENGES AND RECOMMENDATIONS

While this review indicates that IR&A is a growing concern among RT researchers, it is still only being addressed with any depth on a minority of secondary studies. It is clear that several authors believe IR&A is a challenge worth addressing in research, but there is not a lot of available RT literature focusing on the steps that need to be taken in order to improve academia-industry communication and shorten the technology transfer gap.

We conclude this work by highlighting some key challenges that we have identified, combining data found in the literature itself, in the authors' responses and in the practitioner survey. These are challenges that may have been addressed in certain circumstances but remain unsolved in a broad sense, as they are still present in several recent works. Along with each challenge, we make some suggestions that could be applied by Software Engineering researchers and/or Software Testing practitioners — these could be actionable steps for upcoming primary studies, or further avenues of investigation for secondary or meta studies. Table 7 provides the summary of the challenges we identified, indicating the primary source of our observation (i.e. the literature, the authors and/or the practitioners).

CH1: Alignment of motivations. When asked what would convince them to implement and use an RT tool, eight practitioners gave responses that can be synthesized into "*it would make my work easier*". So there exists

a mismatch between academic motivations and industrial needs: research is concerned with discovering novel techniques that might provide marginal effectiveness gains over the state-of-the-art, while practitioners are mostly concerned with any solution that simplifies their workflow. In other words, even if a TCS technique has the potential to greatly reduce the testing time of a suite, practitioners will weigh those benefits against the effort required to implement the technique and adapt/maintain it for their needs. This is not to say that the current research motivations are ill-informed: it is the role of academia to push the boundaries of what is possible in theory first, and sometimes this theory takes many years to find relevance in practice.

If the researchers have the objective of implementing their approach, they must be certain that it is addressing the current needs of practitioners. An obvious way to achieve this, which is also confirmed by our study, is through partnerships between academic researchers and industrial practitioners (or even open-source communities). These collaborative works, by their own nature, tend to produce results suitable for practical applications and could serve as a guideline for other, purely academic, approaches.

Naturally, not all research can be done with industrial partnerships, and in these cases there is difficulty in finding what exactly is relevant to current practitioners. One possible source of this information is grey literature: information produced by experts in a field, but without necessarily following academic guidelines, in the form of blog posts, videos, magazine articles, talks etc. Practitioners who produce grey literature can help inform researchers about the current state of practice, the main existing challenges in software development, and successful implementations of techniques (e.g., the aforementioned Netflix blog [57]).

CH2: Realistic experimentation. It is clearly not possible for every research paper to feature practitioner co-authors or to rely on an industrial partnership for experimentations. Selecting the right subject for experiments is a decisive point when writing a paper about a technique. Older studies on RT would often rely on the "Siemens programs" [53], which is believed to have caused an overfitting of results to a particular kind of software [27]. More recently, the Software Infrastructure Repository (SIR) [28] (e.g., [S3]) and Defects4J [54] (e.g., [S2, S39]) have been used to similar ends. Having common subjects can provide replicability benefits when directly comparing techniques, although is not always clear if they approximate the difficulty of testing real software. Authors who are able to collaborate directly with members of industry gain an enormous advantage if they are allowed to run experiments on production code, but it is also clear that not every paper will have that opportunity.

The most obvious alternative is to use large-scale open-source software (e.g. from the Mozilla [S60] and Apache [S13, S62, S67, S65, S73] foundations) as subjects, since the communities developing these programs follow procedures much like the developers working for corporations. This is also far from trivial. The larger the software, the more time a researcher will need to dedicate in order to understand it and to adapt the technique to it, sacrificing the possibility of experimenting on a larger variety of subjects and thus again bringing the risk of overfitting. Additionally, there is no established consensus regarding which properties an open-source program must satisfy in order to be a satisfactory subject.

Alleviating this issue would require effort from both researchers and practitioners. For example, Google has an open dataset of testing results [31], and S25 combined it with one from ABB Robotics. As a result, this combined dataset has already been used by other papers covering machine learning [S53, S59, S67, S71, S78]. Two practitioners mention that "*open source code/data is not provided*" due to confidentiality reasons. In those cases, our suggestion would be to provide some opaque information regarding the system, such as its programming language, the number of lines of code and/or tests, how many developers work on it, how frequently is the code updated, etc. At the very least, this would help researchers choose subjects with similar characteristics.

CH3: Scalability. RT techniques provide the most savings when applied to large-scale software projects, which can have multiple thousands of test cases. Therefore, it is important that techniques are designed to scale up to any size of test suite, but few papers tackle this issue directly. The trouble is that scalability is very hard to measure unless multiple subjects of different sizes are used. One way to demonstrate scalability, beyond relying

on industrial partners or large-scale open-source projects, is to artificially generate large datasets (e.g, [S34, S50]), which are useful from the algorithmic perspective, but might not address other issues that arise in large-scale software development. It is also worth mentioning that many RT techniques can become *disadvantageous* when applied to small test suites, as the cost of running the technique does not outweigh the savings in testing time. So selecting the size of the experiment subject is important both to highlight the scalability of the tool in large software and also to consider whether the necessary overhead is a deal-breaker on small or medium projects.

CH4: Relevance of metrics. Section 4 shows that a wide variety of metrics has been used to evaluate the effectiveness of RT techniques. Some are used almost universally for a certain kind of challenge (e.g., APFD for TCP), while others have nearly no presence beyond the paper that introduced them.

The abundant use of APFD and its variants indicate that, at least among researchers, there is a consensus of its utility and importance when evaluating TCP approaches, although the usage of specific variants might hamper that benefit. At the same time, it is not clear that a technique optimized for only APFD is sufficient to satisfy the needs of software developers in practice. Still, APFD has been in use for over 20 years and it cannot simply be dismissed: at the very least it provides an agreed-upon method of directly comparing different techniques.

For the cases of TCS and TSR, there is less controversy on what are the most important metrics; reduction rate and fault detection loss appear to be the consensus among researchers, and there are fewer novel and single-use metrics. As an example, S68 interviewed practitioners at Microsoft before deciding on their TCS metrics, obtaining three main targets: reduction of cost, reduction of time and the failure detection rate. We can observe in Section 4 that these concerns are reasonably addressed by TCS techniques, although researchers still appear to prioritize reducing the selected set rather than ensuring all failures are detected.

The metrics of applicability and diagnosability [S46, S60] are interesting propositions that consider other degrees of usefulness of a tool to developers. Their existence indicates that some researchers still believe there is room for improved metrics that, perhaps, better map the requirements of real-world software, although these are rarely found in the literature. Furthermore, ease-of-use is an important point to consider and, as far as we could detect, there is no established method of measuring it.

One practitioner stated: "*I don't think that academic tools are the best in a professional environment, I prefer commercial tools,*" implying they believe academics are not measuring the results that matter most to them. Indeed, managers allocating development funds will usually focus on the dollar savings a technique can bring, regardless of its theoretical effectiveness in fault-finding (as mentioned by respondent author #43).

CH5: Converting research into usable tools. When techniques are designed in an academic context, they are normally developed as proof-of-concept works. That is, the purpose is to show that the technique works and provides significant results according to some metrics. However, this leads to two issues: either primary studies do not make their solution available for implementation, as we discussed in Section 5, or their experiments do not thoroughly consider practical concerns such as efficiency or the data requirements of a proposed approach. Finally, what seems to matter the most is time and budget for developing a tool. Papers are usually written targeting a hard deadline and their prototypes often do not see further work past publication. It is inevitable that researchers will move on to new challenges, but their contribution would be amplified if the tool is, at the very least, open-source and well-documented so that other interested parties can continue the work in the future if desired.

If an RT technique is implemented as a prototype that is shown to work on a certain kind of software, it is much easier to get the attention from a practitioner and convert the solution into something used in practice. If feasible, an available prototype with solid documentation and usage instructions can be valuable both for study replicability and as a way to get developers interested in using it. That said, the responsibility of developing fully functional tools should not fall solely upon researchers. One practitioner stated that "*[RT tools] need full security screening*", and other said "*it requires an adaptation*"; these steps are not actionable by researchers in isolation.

As industry stands to benefit from scientific advances, it should be in its best interest to promote and fund the collaborations needed to continue development of promising prototypes.

CH6: Absence of TSR/TSA. Out of 79 papers, only 8 are about TSR and, surprisingly, only one covers TSA [S17]. 60% of the surveyed practitioners claim that "creating or updating tests" is a major challenge in real-world RT, so the desire for TSA exists and there appears to be ample room for experimenting with new approaches and metrics. 47% also mention the difficulty of refactoring and removing obsolete test cases as a pain point, which is something TSR could remedy. This can be an opportunity for researchers to develop novel methods and to progress in directions that are in need of exploration.

CH7: Clarity of target. Several of the papers we reviewed don't clearly state key characteristics of their SUT, such as its programming language or its scale (either in lines of code or test cases). For practitioners and other researchers to consider a paper worthy of investigation, it is important to know for which kind of system a piece of research was designed.

As mentioned in Section 5, few RT techniques are language-agnostic and many do not inform the target language at all. Similarly, the type of software (web, mobile, embedded, distributed, etc.) or its development paradigm are important factors to mention, seen in studies such as S41 for web services and S59 for software developed and delivered through continuous integration. Not every tool can be used in any type of software, and it is likely that specific types of software might require specific solutions, so it is important to state the particularities of certain subject programs. This is akin to the point of "context factors" brought up by bin Ali et al. [11], which helps to alleviate the issue by introducing a base taxonomy that can be used to categorize techniques.

Critically, there is often ambiguity on the very definition of test case. Software testing can include unit tests, integration tests, multi-component tests, system tests, end-to-end tests and so forth. Most papers do not make it explicit which layer of testing it is addressing. While it can sometimes be inferred with some domain knowledge, it is difficult to be certain for most readers. This information would be valuable for interested practitioners and also for researchers who are looking to identify gaps in the literature. On top of that, some papers use the term "test case" to refer to test methods, while others use it when referring to test classes/files (which contain several test methods), so the granularity of the technique is not always clear, and this can impact both effectiveness and efficiency analysis. This challenge can be solved by having a paragraph dedicated to explicitly describing the properties and context factors of the experiment subjects.

CH8: Communication. The main challenge, which connects most of the previous ones, is communication. Researchers and practitioners both lead busy lives, focusing on their day-to-day affairs, and ultimately communication between the two realms suffers.

There are some steps that can be taken to improve this. Companies can start by having round-table discussions on recent research publications (e.g., the Google Journal Club [81]) and, if possible, they should invite the author(s) to participate. On the other side, universities can host lectures by practitioners in addition to other researchers. This can start small — find people in the same city, perhaps alumni of the university, who are working on something interesting and have a conversation.

56% of responding practitioners claimed they keep contact with a friend or colleague who is a researcher in Software Engineering. After all, most academics have interacted with people who are currently practitioners during their education process, and vice-versa. This means that both sides have an opportunity to network and communicate beyond their current professions, giving each other ideas of what is currently relevant in industrial software development and what is the latest state-of-the-art in academic research.

It can be a daunting idea to catch up to latest research trends, so larger companies could consider having employees dedicated to understanding the internal processes and challenges while searching for collaborations with academics. Many researchers would be thrilled to receive a message inviting them for a joint effort with palpable outcomes.

REFERENCES

[1] Mohamed Abdelkarim and Reem ElAdawi. 2022. TCP-Net: Test Case Prioritization using End-to-End Deep Neural Networks. In *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, Valencia, Spain, 122–129. https://doi.org/10.1109/ICSTW55395.2022.00034

[2] Muhammad Syafiq Abdul Manan, Dayang Norhayati Abang Jawawi, and Johanna Ahmad. 2021. A Systematic Literature Review on Test Case Prioritization in Combinatorial Testing. In *2021 The 5th International Conference on Algorithms, Computing and Systems*. 55–61. https://doi.org/10.1145/3490700.3490710

[3] Farrukh Shahzad Ahmed, Awais Majeed, Tamim Ahmed Khan, and Shahid Nazir Bhatti. 2022. Value-based cost-cognizant test case prioritization for regression testing. *Plos one* 17, 5 (2022). https://doi.org/10.1371/journal.pone.0264972

[4] H. Aman, Y. Tanaka, T. Nakano, H. Ogasawara, and M. Kawahara. 2016. Application of Mahalanobis-Taguchi Method and 0-1 Programming Method to Cost-Effective Regression Testing. In *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*. 240–244. https://doi.org/10.1109/SEAA.2016.29

[5] Maral Azizi and Hyunsook Do. 2018. ReTEST: A Cost Effective Test Case Selection Technique for Modern Software Development. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, Memphis, TN, 144–154. https://doi.org/10.1109/ISSRE.2018.00025

[6] T. Bach, A. Andrzejak, and R. Pannemans. 2017. Coverage-Based Reduction of Test Execution Time: Lessons from a Very Large Industrial Project. In *Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2017*. 3–12. https://doi.org/10.1109/ICSTW.2017.6

[7] Mojtaba Bagherzadeh, Nafiseh Kahani, and Lionel Briand. 2022. Reinforcement Learning for Test Case Prioritization. *IEEE Transactions on Software Engineering* 48, 8 (Aug. 2022), 2836–2856. https://doi.org/10.1109/TSE.2021.3070549

[8] Anu Bajaj and Om Prakash Sangwan. 2018. A Survey on Regression Testing Using Nature-Inspired Approaches. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*. IEEE, Greater Noida, India, 1–5. https://doi.org/10.1109/CCAA.2018.8777692

[9] Anu Bajaj and Om Prakash Sangwan. 2019. A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms. *IEEE Access* 7 (2019), 126355–126375. https://doi.org/10.1109/ACCESS.2019.2938260

[10] Antonia Bertolino, Antonio Guerriero, Breno Miranda, Roberto Pietrantuono, and Stefano Russo. 2020. Learning-to-rank vs ranking-to-learn: strategies for regression testing in continuous integration. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ACM, Seoul South Korea, 1–12. https://doi.org/10.1145/3377811.3380369

[11] Nauman bin Ali, Emelie Engström, Masoumeh Taromirad, Mohammad Reza Mousavi, Nasir Mehmood Minhas, Daniel Helgesson, Sebastian Kunze, and Mahsa Varshosaz. 2019. On the search for industry-relevant regression testing research. *Empirical Software Engineering* 24, 4 (Aug. 2019), 2020–2055. https://doi.org/10.1007/s10664-018-9670-1

[12] Vincent Blondeau, Anne Etien, Nicolas Anquetil, Sylvain Cresson, Pascal Croisy, and Stéphane Ducasse. 2017. Test case selection in industry: an analysis of issues related to static approaches. *Software Quality Journal* 25, 4 (Dec. 2017), 1203–1237. https://doi.org/10.1007/s11219-016-9328-4

[13] G. Buchgeher, C. Klammer, W. Heider, M. Schuetz, and H. Huber. 2016. Improving testing in an enterprise SOA with an architecture-based approach. In *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016*. 231–240. https://doi.org/10.1109/WICSA.2016.24

[14] B. Busjaeger and T. Xie. 2016. Learning for test prioritization: An industrial case study. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Vol. 13-18-November-2016. 975–980. https://doi.org/10.1145/2950290.2983954

[15] Cagatay Catal. 2012. On the application of genetic algorithms for test case prioritization: a systematic literature review. In *Proceedings of the 2nd international workshop on evidential assessment of software technologies*. 9–14. https://doi.org/10.1145/2372233.2372238

[16] Cagatay Catal and Deepti Mishra. 2013. Test case prioritization: a systematic mapping study. *Software Quality Journal* 21, 3 (2013), 445–478. https://doi.org/10.1007/s11219-012-9181-z

[17] Ahmet Celik, Young Chul Lee, and Milos Gligoric. 2018. Regression Test Selection for TizenRT. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. ACM, New York, NY, USA, 845–850. https://doi.org/10.1145/3236024.3275527

[18] A. Celik, M. Vasic, A. Milicevic, and M. Gligoric. 2017. Regression test selection across JVM boundaries. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Vol. Part F130154. 809–820. https://doi.org/10.1145/3106237.3106297

[19] Junjie Chen, Yiling Lou, Lingming Lu Zhang, Jianyi Zhou, Xiaoleng Wang, Dan Hao, and Lingming Lu Zhang. 2018. Optimizing Test Prioritization via Test Distribution Analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. ACM, New York, NY, USA, 656–667. https://doi.org/10.1145/3236024.3236053

[20] Yizhen Chen, Ninad Chaudhari, and Mei-Hwa Chen. 2021. Context-Aware Regression Test Selection. In *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, Taipei, Taiwan, 431–440. https://doi.org/10.1109/APSEC53868.2021.00050

[21] Yizhen Chen and Mei-Hwa Chen. 2021. Multi-Objective Regression Test Selection. 105–92. https://doi.org/10.29007/7z5n

[22] Zongzheng Chi, Jifeng Xuan, Zhilei Ren, Xiaoyuan Xie, and He Guo. 2017. Multi-Level Random Walk for Software Test Suite Reduction. *IEEE Computational Intelligence Magazine* 12, 2 (May 2017), 24–33. https://doi.org/10.1109/MCI.2017.2670460

[23] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46. https://doi.org/10.1177/001316446002000104

[24] D. Correia, R. Abreu, P. Santos, and J. Nadkarni. 2019. MOTSD: A multi-objective test selection tool using test suite diagnosability. In *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1070–1074. https://doi.org/10.1145/3338906.3341187

[25] Emilio Cruciani, Breno Miranda, Roberto Verdecchia, and Antonia Bertolino. 2019. Scalable Approaches for Test Suite Reduction. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, Montreal, QC, Canada, 419–429. https://doi.org/10.1109/ICSE.2019.00055

[26] Benjamin Danglot, Oscar Vera-Perez, Zhongxing Yu, Andy Zaidman, Martin Monperrus, and Benoit Baudry. 2019. A snowballing literature study on test amplification. *Journal of Systems and Software* 157 (2019), 110398. https://doi.org/10.1016/j.jss.2019.110398

[27] Hyunsook Do. 2016. Recent Advances in Regression Testing Techniques. In *Advances in Computers*. Vol. 103. Elsevier, 53–77. https://doi.org/10.1016/bs.adcom.2016.04.004

[28] Hyunsook Do, Sebastian Elbaum, and Gregg Rothermel. 2005. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering* 10, 4 (2005), 405–435. https://doi.org/10.1007/s10664-005-3861-2

[29] Ivan do Carmo Machado, John D McGregor, Yguaratã Cerqueira Cavalcanti, and Eduardo Santana De Almeida. 2014. On strategies for testing software product lines: A systematic literature review. *Information and Software Technology* 56, 10 (2014), 1183–1199. https://doi.org/10.1016/j.infsof.2014.04.002

[30] Ravi Eda and Hyunsook Do. 2019. An efficient regression testing approach for PHP Web applications using test selection and reusable constraints. *Software Quality Journal* 27, 4 (Dec. 2019), 1383–1417. https://doi.org/10.1007/s11219-019-09449-2

[31] Sebastian Elbaum, Andrew Mclaughlin, and John Penix. 2014. The Google Dataset of Testing Results. https://code.google.com/p/google-shared-dataset-of-test-suite-results

[32] Daniel Elsner, Florian Hauer, Alexander Pretschner, and Silke Reimer. 2021. Empirically evaluating readily available information for regression test optimization in continuous integration. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, Virtual Denmark, 491–504. https://doi.org/10.1145/3460319.3464834

[33] Emelie Engström and Per Runeson. 2010. A qualitative survey of regression testing practices. In *International Conference on Product Focused Software Process Improvement*. Springer, 3–16. https://doi.org/10.1007/978-3-642-13792-1_3

[34] Emelie Engström, Per Runeson, and Mats Skoglund. 2010. A systematic review on regression test selection techniques. *Information and Software Technology* 52, 1 (2010), 14–30. https://doi.org/10.1016/j.infsof.2009.07.001

[35] Michael Felderer, Matthias Büchler, Martin Johns, Achim D. Brucker, Ruth Breu, and Alexander Pretschner. 2016. Chapter One - Security Testing: A Survey. Advances in Computers, Vol. 101. Elsevier, 1–51. https://doi.org/10.1016/bs.adcom.2015.11.003

[36] Michael Felderer and Elizabeta Fourneret. 2015. A systematic classification of security regression testing approaches. *International Journal on Software Tools for Technology Transfer* 17, 3 (2015), 305–319. https://doi.org/10.1007/s10009-015-0365-2

[37] Ben Fu, Sasa Misailovic, and Milos Gligoric. 2019. Resurgence of Regression Test Selection for C++. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. IEEE, Xi'an, China, 323–334. https://doi.org/10.1109/ICST.2019.00039

[38] Paul Garner, Sally Hopewell, Jackie Chandler, Harriet MacLehose, Elie A Akl, Joseph Beyene, Stephanie Chang, Rachel Churchill, Karin Dearness, Gordon Guyatt, Carol Lefebvre, Beth Liles, Rachel Marshall, Laura Martínez García, Chris Mavergames, Mona Nasser, Amir Qaseem, Margaret Sampson, Karla Soares-Weiser, Yemisi Takwoingi, Lehana Thabane, Marialena Trivella, Peter Tugwell, Emma Welsh, Ed C Wilson, and Holger J Schünemann. 2016. When and how to update systematic reviews: consensus and checklist. *BMJ* 354 (2016). https://doi.org/10.1136/bmj.i3507

[39] Vahid Garousi and Michael Felderer. 2017. Worlds apart: industrial and academic focus areas in software testing. *IEEE Software* 34, 5 (2017), 38–45. https://doi.org/10.1109/MS.2017.3641116

[40] V. Garousi, R. Özkan, and A. Betin-Can. 2018. Multi-objective regression test selection in practice: An empirical study in the defense software industry. *Information and Software Technology* 103 (2018), 40–54. https://doi.org/10.1016/j.infsof.2018.06.007

[41] A. Gotlieb and D. Marijan. 2017. Using global constraints to automate regression testing. *AI Magazine* 38, 1 (2017), 73–87. https://doi.org/10.1609/aimag.v38i1.2714

[42] A. Goyal, R.K. Shyamasundar, R. Jetley, D. Mohan, and S. Ramaswamy. 2019. Test suite minimization of evolving software systems: A case study. In *ICSOFT 2019 - Proceedings of the 14th International Conference on Software Technologies*. 226–237. https://doi.org/10.5220/0007842502260237

[43] Renan Greca, Breno Miranda, Milos Gligoric, and Antonia Bertolino. 2022. Comparing and combining file-based selection and similarity-based prioritization towards regression test orchestration. In *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test*. ACM, Pittsburgh Pennsylvania, 115–125. https://doi.org/10.1145/3524481.3527223

[44] B. Guo, Y.-W. Kwon, and M. Song. 2019. Decomposing Composite Changes for Code Review and Regression Test Selection in Evolving Software. *Journal of Computer Science and Technology* 34, 2 (2019), 416–436. https://doi.org/10.1007/s11390-019-1917-9

[45] Alireza Haghighatkhah, Mika Mäntylä, Markku Oivo, and Pasi Kuvaja. 2018. Test prioritization in continuous integration environments. *Journal of Systems and Software* 146 (Dec. 2018), 80–98. https://doi.org/10.1016/j.jss.2018.08.061

[46] Dan Hao, Lu Zhang, and Hong Mei. 2016. Test-case prioritization: achievements and challenges. *Frontiers of Computer Science* 10, 5 (Oct. 2016), 769–777. https://doi.org/10.1007/s11704-016-6112-3

[47] Mary Jean Harrold and Alessandro Orso. 2008. Retesting software during development and maintenance. In *2008 Frontiers of Software Maintenance*. IEEE, 99–108. https://doi.org/10.1109/FOSM.2008.4659253

[48] Muhammad Hasnain, Imran Ghani, Muhammad Fermi Pasha, and Seung Ryul Jeong. 2020. A Comprehensive Review on Regression Test Case Prioritization Techniques for Web Services. *KSII Transactions on Internet and Information Systems* 14, 5 (May 2020). https://doi.org/10.3837/tiis.2020.05.001

[49] Muhammad Hasnain, Imran Ghani, Muhammad Fermi Pasha, and Seung-Ryul Jeong. 2021. Ontology-Based Regression Testing: A Systematic Literature Review. *Applied Sciences* 11, 20 (2021), 9709. https://doi.org/10.3390/app11209709

[50] Kim Herzig. 2016. Let's assume we had to pay for testing. https://www.slideshare.net/kim.herzig/keynote-ast-2016

[51] Kim Herzig, Michaela Greiler, Jacek Czerwonka, and Brendan Murphy. 2015. The Art of Testing Less without Sacrificing Quality. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (Florence, Italy) *(ICSE '15)*. IEEE Press, 483–493. https://doi.org/10.1109/ICSE.2015.66

[52] Matthias Hirzel and Herbert Klaeren. 2016. Graph-Walk-based Selective Regression Testing of Web Applications Created with Google Web Toolkit. (2016), 15.

[53] Monica Hutchins, Herb Foster, Tarak Goradia, and Thomas Ostrand. 1994. Experiments on the effectiveness of dataflow-and control-flow-based test adequacy criteria. In *Proceedings of 16th International conference on Software engineering*. IEEE, 191–200. https://doi.org/10.1109/ICSE.1994.296778

[54] René Just, Darioush Jalali, and Michael D Ernst. 2014. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. 437–440. https://doi.org/10.1145/2610384.2628055

[55] Rafaqut Kazmi, Dayang N. A. Jawawi, Radziah Mohamad, and Imran Ghani. 2017. Effective Regression Test Case Selection: A Systematic Literature Review. *Comput. Surveys* 50, 2 (June 2017), 1–32. https://doi.org/10.1145/3057269

[56] Muhammad Khatibsyarbini, Mohd Adham Isa, Dayang N.A. Jawawi, and Rooster Tumeng. 2018. Test case prioritization approaches in regression testing: A systematic literature review. *Information and Software Technology* 93 (Jan. 2018), 74–93. https://doi.org/10.1016/j.infsof.2017.08.014

[57] Stanislav Kirdey, Kevin Cureton, Scott Rick, Sankar Ramanathan, and Mrinal Shukla. 2019. Lerner — using RL agents for test case scheduling. https://netflixtechblog.com/lerner-using-rl-agents-for-test-case-scheduling-3e0686211198

[58] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.

[59] Jung-Hyun Kwon and In-Young Ko. 2017. Cost-Effective Regression Testing Using Bloom Filters in Continuous Integration Development Environments. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, Nanjing, 160–168. https://doi.org/10.1109/APSEC.2017.22

[60] Adriaan Labuschagne, Laura Inozemtseva, and Reid Holmes. 2017. Measuring the Cost of Regression Testing in Practice: A Study of Java Projects Using Continuous Integration. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn, Germany) *(ESEC/FSE 2017)*. ACM, New York, NY, USA, 821–830. https://doi.org/10.1145/3106237.3106288

[61] Kathrin Land, Eva-Maria Neumann, Simon Ziegltrum, Huaxia Li, and Birgit Vogel-Heuser. 2019. An Industrial Evaluation of Test Prioritisation Criteria and Metrics. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, Bari, Italy, 1887–1892. https://doi.org/10.1109/SMC.2019.8914505

[62] Claire Leong, Abhayendra Singh, Mike Papadakis, Yves Le Traon, and John Micco. 2019. Assessing Transition-Based Test Selection Algorithms at Google. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, Montreal, QC, Canada, 101–110. https://doi.org/10.1109/ICSE-SEIP.2019.00019

[63] Hareton KN Leung and Lee White. 1989. Insights into regression testing (software testing). In *Proceedings. Conference on Software Maintenance-1989*. IEEE, 60–69. https://doi.org/10.1109/ICSM.1989.65194

[64] Feng Li, Jianyi Zhou, Yinzhu Li, Dan Hao, and Lu Zhang. 2021. AGA: An Accelerated Greedy Additional Algorithm for Test Case Prioritization. *IEEE Transactions on Software Engineering* (2021), 1–1. https://doi.org/10.1109/TSE.2021.3137929

[65] Jackson A. Prado Lima and Silvia Regina Vergilio. 2022. A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments. *IEEE Transactions on Software Engineering* 48, 2 (Feb. 2022), 453–465. https://doi.org/10.1109/TSE.2020.2992428

[66] Yiling Lou, Junjie Chen, Lingming Zhang, and Dan Hao. 2019. A Survey on Regression Test-Case Prioritization. *Advances in Computers* 113 (2019), 1–46. https://doi.org/10.1016/bs.adcom.2018.10.001

[67] Yafeng Lu, Yiling Lou, Shiyang Cheng, Lingming Zhang, Dan Hao, Yangfan Zhou, and Lu Zhang. 2016. How does regression test prioritization perform in real-world software evolution?. In *Proceedings of the 38th International Conference on Software Engineering*.

ACM, Austin Texas, 535–546. https://doi.org/10.1145/2884781.2884874

[68] Daniel Lübke. 2020. Selecting and Prioritizing Regression Test Suites by Production Usage Risk in Time-Constrained Environments. (2020), 69–83. https://doi.org/10.1007/978-3-030-35510-4

[69] Mateusz Machalica, Alex Samylkin, Meredith Porth, and Satish Chandra. 2019. Predictive Test Selection. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 91–100. https://doi.org/10.1109/ICSE-SEIP.2019.00018

[70] Cláudio Magalhães, Flávia Barros, Alexandre Mota, and Eliot Maia. 2016. Automatic Selection of Test Cases for Regression Testing. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST*. ACM Press, Maringa, Parana, Brazil, 1–8. https://doi.org/10.1145/2993288.2993299

[71] Claudio Magalhães, João Andrade, Lucas Perrusi, Alexandre Mota, Flávia Barros, and Eliot Maia. 2020. HSP: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study. *Journal of Systems and Software* 159 (Jan. 2020), 110430. https://doi.org/10.1016/j.jss.2019.110430

[72] Dusica Marijan and Marius Liaaen. 2016. Effect of Time Window on the Performance of Continuous Regression Testing. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Raleigh, NC, USA, 568–571. https://doi.org/10.1109/ICSME.2016.77

[73] Sonu Mehta, Farima Farmahinifarahani, Ranjita Bhagwan, Suraj Guptha, Sina Jafari, Rahul Kumar, Vaibhav Saini, and Anirudh Santhiar. 2021. Data-driven test selection at scale. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Athens Greece, 1225–1235. https://doi.org/10.1145/3468264.3473916

[74] Atif Memon, Zebao Gao, Bao Nguyen, Sanjeev Dhanda, Eric Nickell, Rob Siemborski, and John Micco. 2017. Taming Google-scale continuous testing. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 233–242. https://doi.org/10.1109/ICSE-SEIP.2017.16

[75] Emilia Mendes, Claes Wohlin, Katia Felizardo, and Marcos Kalinowski. 2020. When to update systematic literature reviews in software engineering. *Journal of Systems and Software* 167 (2020), 110607. https://doi.org/10.1016/j.jss.2020.110607

[76] Breno Miranda, Emilio Cruciani, Roberto Verdecchia, and Antonia Bertolino. 2018. FAST approaches to scalable similarity-based test case prioritization. In *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*. ACM Press, Gothenburg, Sweden, 222–232. https://doi.org/10.1145/3180155.3180210

[77] Muhammad Luqman Mohd-Shafie, Wan Mohd Nasir Wan Kadir, Horst Lichter, Muhammad Khatibsyarbini, and Mohd Adham Isa. 2021. Model-based test case generation and prioritization: a systematic literature review. *Software and Systems Modeling* (2021), 1–37. https://doi.org/10.1007/s10270-021-00924-8

[78] Rajendrani Mukherjee and K. Sridhar Patnaik. 2018. A survey on different approaches for software test case prioritization. *Journal of King Saud University - Computer and Information Sciences* (Oct. 2018), S1319157818303616. https://doi.org/10.1016/j.jksuci.2018.09.005

[79] A. Najafi, W. Shang, and P.C. Rigby. 2019. Improving Test Effectiveness Using Test Executions History: An Industrial Experience Report. In *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*. 213–222. https://doi.org/10.1109/ICSE-SEIP.2019.00031

[80] Tutku Çıngıl and Hasan Sözer. 2022. Black-box Test Case Selection by Relating Code Changes with Previously Fixed Defects. In *The International Conference on Evaluation and Assessment in Software Engineering 2022*. ACM, Gothenburg Sweden, 30–39. https://doi.org/10.1145/3530019.3530023

[81] Bao N. Nguyen, Tim Henderson, John Micco, and Sanjeev Dhanda. 2016. Google Journal Club. https://sites.google.com/site/gjournalclub/

[82] R. Noemmer and R. Haas. 2020. An Evaluation of Test Suite Minimization Techniques. *Lecture Notes in Business Information Processing* 371 LNBIP (2020), 51–66. https://doi.org/10.1007/978-3-030-35510-4_4

[83] T.B. Noor and H. Hemmati. 2016. A similarity-based approach for test case prioritization using historical failure data. In *2015 IEEE 26th International Symposium on Software Reliability Engineering, ISSRE 2015*. 58–68. https://doi.org/10.1109/ISSRE.2015.7381799

[84] Safa Omri and Carsten Sinz. 2022. Learning to Rank for Test Case Prioritization. In *2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)*. 16–24. https://doi.org/10.1145/3526072.3527525

[85] João Felipe S. Ouriques, Emanuela G. Cartaxo, and Patrícia D.L. Machado. 2018. Test case prioritization techniques for model-based testing: a replicated study. *Software Quality Journal* 26, 4 (Dec. 2018), 1451–1482. https://doi.org/10.1007/s11219-017-9398-y

[86] Chaoyue Pan, Yang Yang, Zheng Li, and Junxia Guo. 2020. Dynamic Time Window based Reward for Reinforcement Learning in Continuous Integration Testing. In *12th Asia-Pacific Symposium on Internetware*. ACM, Singapore Singapore, 189–198. https://doi.org/10.1145/3457913.3457930

[87] Rongqi Pan, Mojtaba Bagherzadeh, Taher A Ghaleb, and Lionel Briand. 2022. Test case selection and prioritization using machine learning: a systematic literature review. *Empirical Software Engineering* 27, 2 (2022), 1–43. https://doi.org/10.1007/s10664-021-10066-6

[88] Qianyang Peng, August Shi, and Lingming Zhang. 2020. Empirically revisiting and enhancing IR-based test-case prioritization. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, Virtual Event USA, 324–336. https://doi.org/10.1145/3395363.3397383

[89] Adithya Abraham Philip, Ranjita Bhagwan, Rahul Kumar, Chandra Sekhar Maddila, and Nachiappan Nagppan. 2019. FastLane: Test Minimization for Rapidly Deployed Large-Scale Online Services. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, Montreal, QC, Canada, 408–418. https://doi.org/10.1109/ICSE.2019.00054

[90] Dipesh Pradhan, Shuai Wang, Shaukat Ali, and Tao Yue. 2016. Search-Based Cost-Effective Test Case Selection within a Time Budget: An Empirical Study. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, New York, NY, USA, 1085–1092. https://doi.org/10.1145/2908812.2908850

[91] Jackson A. Prado Lima and Silvia R. Vergilio. 2020. Test Case Prioritization in Continuous Integration environments: A systematic mapping study. *Information and Software Technology* 121 (2020). https://doi.org/10.1016/j.infsof.2020.106268

[92] Dong Qiu, Bixin Li, Shunhui Ji, and Hareton Leung. 2014. Regression testing of web service: a systematic mapping study. *ACM Computing Surveys (CSUR)* 47, 2 (2014), 1–46. https://doi.org/10.1145/2631685

[93] Jesper Öqvist, Görel Hedin, and Boris Magnusson. 2016. Extraction-Based Regression Test Selection. In *Proceedings of the 13th International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*. ACM, Lugano Switzerland, 1–10. https://doi.org/10.1145/2972206.2972224

[94] R. Ramler, C. Salomon, G. Buchgeher, and M. Lusser. 2017. Tool support for change-based regression testing: An industry experience report. *Lecture Notes in Business Information Processing* 269 (2017), 133–152. https://doi.org/10.1007/978-3-319-49421-0_10

[95] Saif Ur Rehman Khan, Sai Peck Lee, Nadeem Javaid, and Wadood Abdul. 2018. A Systematic Review on Test Suite Reduction: Approaches, Experiment's Quality Evaluation, and Guidelines. *IEEE Access* 6 (2018), 11816–11841. https://doi.org/10.1109/ACCESS.2018.2809600

[96] Lukas Rosenbauer, Anthony Stein, and Jörg Hähner. 2021. An Artificial Immune System for Black Box Test Case Selection. In *Evolutionary Computation in Combinatorial Optimization*, Christine Zarges and Sébastien Verel (Eds.). Vol. 12692. Springer International Publishing, Cham, 169–184. https://doi.org/10.1007/978-3-030-72904-2_11

[97] Raúl H. Rosero, Omar S. Gómez, and Glen Rodríguez. 2016. 15 Years of Software Regression Testing Techniques - A Survey. *International Journal of Software Engineering and Knowledge Engineering* 26, 5 (2016), 675–689. https://doi.org/10.1142/S0218194016300013

[98] Raúl H Rosero, Omar S Gómez, Eduardo R Villa, Raúl A Aguilar, and César J Pardo. 2021. Software Regression Testing in Industrial Settings: Preliminary Findings from a Literature Review. In *The International Conference on Advances in Emerging Trends and Technologies*. Springer, 227–237. https://doi.org/10.1007/978-3-030-96147-3_18

[99] Zahra Sadri-Moshkenani, Justin Bradley, and Gregg Rothermel. 2022. Survey on test case generation, selection and prioritization for cyber-physical systems. *Software Testing, Verification and Reliability* 32, 1 (2022), e1794. https://doi.org/10.1002/stvr.1794

[100] Ali Samad, Hairulnizam Mahdin, Rafaqut Kazmi, and Rosziati Ibrahim. 2021. Regression Test Case Prioritization: A Systematic Literature Review. *International Journal of Advanced Computer Science and Applications* 12, 2 (2021). https://doi.org/10.14569/IJACSA.2021.0120282

[101] Raúl A. Santelices, Pavan Kumar Chittimalli, Taweesup Apiwattanapong, Alessandro Orso, and Mary Jean Harrold. 2008. Test-Suite Augmentation for Evolving Software. In *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy*. IEEE Computer Society, 218–227. https://doi.org/10.1109/ASE.2008.32

[102] Amanda Schwartz and Hyunsook Do. 2016. Cost-effective regression testing through Adaptive Test Prioritization strategies. *Journal of Systems and Software* 115 (May 2016), 61–81. https://doi.org/10.1016/j.jss.2016.01.018

[103] Aizaz Sharif, Dusica Marijan, and Marius Liaaen. 2021. DeepOrder: Deep Learning for Test Case Prioritization in Continuous Integration Testing. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Luxembourg, 525–534. https://doi.org/10.1109/ICSME52107.2021.00053

[104] August Shi, Alex Gyori, Suleman Mahmood, Peiyuan Zhao, and Darko Marinov. 2018. Evaluating test-suite reduction in real software evolution. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 84–94. https://doi.org/10.1145/3213846.3213875

[105] August Shi, Peiyuan Zhao, and Darko Marinov. 2019. Understanding and Improving Regression Test Selection in Continuous Integration. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, Berlin, Germany, 228–238. https://doi.org/10.1109/ISSRE.2019.00031

[106] Yogesh Singh, Arvinder Kaur, Bharti Suri, and Shweta Singhal. 2012. Systematic literature review on regression test prioritization techniques. *Informatica* 36, 4 (2012).

[107] Helge Spieker, Arnaud Gotlieb, Dusica Marijan, and Morten Mossige. 2017. Reinforcement learning for automatic test case prioritization and selection in continuous integration. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, Santa Barbara CA USA, 12–22. https://doi.org/10.1145/3092703.3092709

[108] H. Srikanth, M. Cashman, and M.B. Cohen. 2016. Test case prioritization of build acceptance tests for an enterprise cloud application: An industrial case study. *Journal of Systems and Software* 119 (2016), 122–135. https://doi.org/10.1016/j.jss.2016.06.017

[109] Hema Srikanth, Charitha Hettiarachchi, and Hyunsook Do. 2016. Requirements Based Test Prioritization Using Risk Factors. *Inf. Softw. Technol.* 69, C (Jan. 2016), 71–83. https://doi.org/10.1016/j.infsof.2015.09.002

[110] P.E. Strandberg, D. Sundmark, W. Afzal, T.J. Ostrand, and E.J. Weyuker. 2016. Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*. 12–23. https://doi.org/10.1109/ISSRE.2016.23

[111] S. Tahvili, M. Bohlin, M. Saadatmand, S. Larsson, W. Afzal, and D. Sundmark. 2016. Cost-benefit analysis of using dependency knowledge at integration testing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10027 LNCS (2016), 268–284. https://doi.org/10.1007/978-3-319-49094-6_17

[112] Sahar Tahvili, Mehrdad Saadatmand, Stig Larsson, Wasif Afzal, Markus Bohlin, and Daniel Sundmark. 2016. Dynamic Integration Test Selection Based on Test Case Dependencies. In *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, Chicago, IL, USA, 277–286. https://doi.org/10.1109/ICSTW.2016.14

[113] Marko Vasic, Zuhair Parvez, Aleksandar Milicevic, and Milos Gligoric. 2017. File-level vs. module-level regression test selection for .NET. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, Paderborn Germany, 848–853. https://doi.org/10.1145/3106237.3117763

[114] Sebastian Vöst and Stefan Wagner. 2016. Trace-based test selection to support continuous integration in the automotive industry. In *Proceedings - International Workshop on Continuous Software Evolution and Delivery, CSED 2016*. ACM, 34–40. https://doi.org/10.1145/2896941.2896951

[115] Shuai Wang, Shaukat Ali, Tao Yue, Oyvind Bakkeli, and Marius Liaaen. 2016. Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search. In *Proceedings - International Conference on Software Engineering*. 182–191. https://doi.org/10.1145/2889160.2889240

[116] Zhaolin Wu, Yang Y. Yang, Zheng Li, and Ruilian Zhao. 2019. A Time Window Based Reinforcement Learning Reward for Test Case Prioritization in Continuous Integration. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware (Internetware '19)*. ACM, New York, NY, USA. https://doi.org/10.1145/3361242.3361258

[117] Jincheng Xu, Qingfeng Du, and Xiaojun Li. 2021. A Requirement-based Regression Test Selection Technique in Behavior-Driven Development. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, Madrid, Spain, 1303–1308. https://doi.org/10.1109/COMPSAC51774.2021.00182

[118] J.J. Yackley, M. Kessentini, G. Bavota, V. Alizadeh, and B.R. Maxim. 2019. Simultaneous refactoring and regression testing. In *Proceedings - 19th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2019*. 216–227. https://doi.org/10.1109/SCAM.2019.00032

[119] Ahmadreza Saboor Yaraghi, Mojtaba Bagherzadeh, Nafiseh Kahani, and Lionel Briand. 2022. Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts. *IEEE Transactions on Software Engineering* (2022), 1–24. https://doi.org/10.1109/TSE.2022.3184842

[120] U. Yilmaz and A. Tarhan. 2018. A case study to compare regression test selection techniques on open-source software projects. In *CEUR Workshop Proceedings*, Vol. 2201.

[121] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability* 22, 2 (2012), 67–120. https://doi.org/10.1002/stvr.430

[122] H. Yoshida, S. Tokumoto, M.R. Prasad, I. Ghosh, and T. Uehara. 2016. FSX: A tool for fine-grained incremental unit test generation for C/C++ Programs. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Vol. 13-18-November-2016. 1052–1056. https://doi.org/10.1145/2950290.2983937

[123] Zhe Yu, Fahmid Fahid, Tim Menzies, Gregg Rothermel, Kyle Patrick, and Snehit Cherian. 2019. TERMINATOR: better automated UI test case prioritization. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Tallinn Estonia, 883–894. https://doi.org/10.1145/3338906.3340448

[124] Anis Zarrad. 2015. A Systematic Review on Regression Testing for Web-Based Applications. *J. Softw.* 10, 8 (2015), 971–990.

[125] Jiyang Zhang, Yu Liu, Milos Gligoric, Owolabi Legunsen, and August Shi. 2022. Comparing and combining analysis-based and learning-based regression test selection. In *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test*. ACM, Pittsburgh Pennsylvania, 17–28. https://doi.org/10.1145/3524481.3527230

[126] Lingming Zhang. 2018. Hybrid regression test selection. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, Gothenburg Sweden, 199–209. https://doi.org/10.1145/3180155.3180198

[127] Hua Zhong, Lingming Zhang, and Sarfraz Khurshid. 2019. TestSage: Regression Test Selection for Large-Scale Web Service Testing. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. IEEE, Xi'an, China, 430–440. https://doi.org/10.1109/ICST.2019.00052

[128] Jianyi Zhou, Junjie Chen, and Dan Hao. 2022. Parallel Test Prioritization. *ACM Transactions on Software Engineering and Methodology* 31, 1 (Jan. 2022), 1–50. https://doi.org/10.1145/3471906

[129] Zhi Quan Zhou, Chen Liu, Tsong Yueh Chen, T. H. Tse, and Willy Susilo. 2020. Beating Random Test Case Prioritization. *IEEE Transactions on Reliability* (2020), 1–22. https://doi.org/10.1109/TR.2020.2979815

[130] Yuecai Zhu, Emad Shihab, and Peter C. Rigby. 2018. Test re-prioritization in continuous testing environments. In *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018*. IEEE, 69–79. https://doi.org/10.1109/ICSME.2018.00016