Mapping the structure and evolution of software testing research over the past three decades

Citation for the original published paper (version of record):
Salahirad, A., Gay, G., Mohammadi, E. (2023). Mapping the structure and evolution of software testing research over the past three decades. Journal of Systems and Software, 195.
http://dx.doi.org/10.1016/j.jss.2022.111518

The Journal of Systems & Software 195 (2023) 111518

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Mapping the structure and evolution of software testing research over the past three decades[☆]

Alireza Salahirad [a], Gregory Gay [b,*], Ehsan Mohammadi [c]

[a] Department of Computer Science & Engineering, University of South Carolina, USA
[b] Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Sweden
[c] School of Information Science, University of South Carolina, USA

ARTICLE INFO

Article history:
Received 9 September 2021
Received in revised form 19 September 2022
Accepted 20 September 2022
Available online 24 September 2022

Keywords:
Software testing
Bibliometrics
Co-word analysis

ABSTRACT

Background: The field of software testing is growing and rapidly-evolving.
Aims: Based on keywords assigned to publications, we seek to identify predominant research topics and understand how they are connected and have evolved.
Methods: We apply co-word analysis to map the topology of testing research as a network where author-assigned keywords are connected by edges indicating co-occurrence in publications. Keywords are clustered based on edge density and frequency of connection. We examine the most popular keywords, summarize clusters into high-level research topics examine how topics connect, and examine how the field is changing.
Results: Testing research can be divided into 16 high-level topics and 18 subtopics. Creation guidance, automated test generation, evolution and maintenance, and test oracles have particularly strong connections to other topics, highlighting their multidisciplinary nature. Emerging keywords relate to web and mobile apps, machine learning, energy consumption, automated program repair and test generation, while emerging connections have formed between web apps, test oracles, and machine learning with many topics. Random and requirements-based testing show potential decline.
Conclusions: Our observations, advice, and map data offer a deeper understanding of the field and inspiration regarding challenges and connections to explore.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Software testing refers to the application of input to a system to identify issues affecting its correctness or its ability to deliver services (Pezze and Young, 2006). While many quality assurance techniques exist, testing remains the primary means of assessing software quality.

From nearly the beginning of software development as a discipline, researchers and practitioners have reasoned about testing and quality assurance (Turing, 1989). Today, testing is one of the largest areas of software engineering research (Orso and Rothermel, 2014), and the field is rapidly evolving as new software and hardware advances are introduced. It is useful, therefore, to understand (a) *what the predominant research topics are of the field*, (b) *how those topics are connected*, and (c), *how the predominant topics have evolved over time*.

"Science of science" describes a research methodology where text, author, and publication metadata are analyzed using quantitative bibliometric and scientometric techniques (Fortunato et al., 2018; Borgman and Furner, 2002). Computational methods, such as text mining and citation analysis, map the topical structure of a research field, enabling the discovery of invisible patterns and relationships in the publications that form that field (Moed, 2006; Ding et al., 2013).

We have applied co-word analysis to visualize and analyze the topology of 35 years of software testing research, based on the author-assigned keywords of Scopus-indexed publications. Co-word analysis yields an undirected network where the nodes – author-assigned keywords – represent targeted research concepts. Weighted edges connect keywords, based on their co-occurrence on publications. Finally, keywords are grouped into clusters, representing densely-connected regions of the network.

Our analysis maps keywords into dense clusters, from which emerge high-level research topics – themes that characterize

☆ Editor: Matthias Galster.

* Corresponding author.
E-mail addresses: alireza@email.sc.edu (A. Salahirad), greg@greggay.com (G. Gay), ehsan2@sc.edu (E. Mohammadi).

https://doi.org/10.1016/j.jss.2022.111518

each cluster – and makes clear the connections between keywords and topics within and across clusters. It also characterizes the periods in which low-level keywords and high-level topics have emerged—identifying emerging research areas, as well as those where research interest has decreased. The goal of this study is to provide both current and future researchers with perspectives about testing field, built on a quantitative base. For researchers, a snapshot of important disciplinary trends can provide valuable insight into the state of the field, suggest topics to explore, and identify connections (or lack thereof) between keywords and topics that may reveal new insights. Among others, we have made the following observations:

- Both the most common author-assigned keywords and the keywords that attract the most citations, on average, tend to relate to automation, test creation and assessment guidance, assessment of system quality, and cyber–physical systems.
- These keywords can be clustered into 16 topics: automated test generation, creation guidance, evolution and maintenance, machine learning and predictive modeling, model-based testing, GUI testing, processes and risk, random testing, reliability, requirements, system testing, test automation, test case types, test oracles, verification and program analysis, and web application testing. Below these lie 18 more subtopics.
- Creation guidance, automated test generation, evolution and maintenance, and test oracles are particularly multidisciplinary topics, with dense connections to many other topics. Twenty keywords connect topics, reflecting multidisciplinary concepts, common test activities, and test creation information.
- Emerging research particularly relates to web and mobile applications, ML and AI – including autonomous vehicles – energy consumption, automated program repair, or fuzzing and search-based test generation. Web applications require targeted testing approaches and practices, leading to emerging connections to many topics. Test oracles are also a rapidly-evolving topic with many emerging connections. ML has emerging potential to support automation.
- Research related to random and requirements-based testing may be in decline.

We believe that these insights – and the rich underlying networks of keywords – can inspire both current and future researchers in the field of software testing. We additionally make our data available so that others may make their own observations or broaden the horizons of their own research.[1]

The remainder of this publication is structured as follows. In Section 2, we discuss background concepts and related work. In Section 3, we explain our methodology. Section 4 answers our research questions. In Section 5, we provide advice on the use of this data, as well as exploratory analyses related to under-explored and missing connections. Section 6 details threats to validity. In Section 7, we offer our conclusions.

2. Background and related work

2.1. Bibliometrics and Co-Word analysis

Bibliometric analysis is "the application of mathematical and statistical methods to books and other means of communication" (Pritchard et al., 1969). Bibliometric studies perform quantitative analysis of publications and associated metadata – e.g., keywords, authors, institutions, and citations – to identify themes and patterns within a research field (De Bellis, 2009). Such analysis is often combined with mapping techniques to visualize hidden structures in the metadata of a particular field (Donthu et al., 2020). The most common analysis methods used include citation-based, co-word (also known as keyword co-occurrence), and co-authorship analysis (Van Eck and Waltman, 2014). We focus on co-word analysis.

In co-word analysis, natural language processing and text mining techniques are used to discover the most meaningful noun phrases in a collection of documents and visualize their meaning in a two-dimensional map (Peters and van Raan, 1993). In this map, co-occurring terms are connected, with "closer" placement resulting from stronger co-occurrence. Co-word analysis is generally based on the number of research publications where two keywords are used together to describe the research performed (Whittaker, 1989). Because keywords succinctly capture the context of a publication, co-word analysis is an effective method of revealing connections between publications (Su and Lee, 2010) and identifying trends in a field (Ding et al., 2013).

Scholars have previously used co-word analysis to depict the structure of fields including renewable energy (Romo-Fernández et al., 2013), global warming (Marx et al., 2017), nanoscience and nanotechnology (Mohammadi, 2012), human computer interaction (Liu et al., 2014), and big data (Liao et al., 2018; Mohammadi and Karami, 2020). Our study is the first to apply such techniques to software testing.

2.2. Bibliometrics and software engineering

Our study is the first to apply scientometric or bibliometric techniques to the software testing field. However, bibliometric techniques have been applied to other aspects of software engineering (SE). In Table 1, we contrast our study to related work. Below, we further elaborate on the specific studies. In general, we focus on analysis of research topics and the connections between topics, and do not analyze authorship trends. Our focus and chosen analysis methods enable a deep characterization of the connections between topics and low-level publication keywords in software testing.

Garousi and Mäntylä performed a bibliometric analysis of more than 70,000 general SE publications, finding that the most popular research topics were web applications, mobile and cloud computing, industrial case studies, source code, and automated test generation (Garousi and Mäntylä, 2016). Our identified research topics include all of these except source code – which is subsumed by other topics – and case studies. In our study, case studies would be categorized based on the problems they address. They also found that a small number of large countries produce the majority of publications, while small European countries are proportionally the most active in the field.

Garousi and Fernandes used the same set of publications to assess questions related to quantity versus the impact of SE research (Garousi and Fernandes, 2017). They broadly found that journal articles have more impact than conference publications and that publications from English-speaking researchers have more visibility and impact. Both studies also used Scopus to gather publications, but had a different focus from our study (all of "software", rather than software testing). The studies also differ in their analysis methods. Rather than co-word analysis, the authors of both studies used citation-based analyses. Co-word analysis allows examination of the connections between topics.

Karanatsiou et al. targeted SE publications from 2010–2017 for analysis, identifying top institutions and scholars from this period (Karanatsiou et al., 2019). Wong et al. did the same for the periods of 2001–2005 (Wong et al., 2008), 2002–2006 (Wong et al., 2009), and 2003–2007 and 2004–2008 (Wong et al., 2011).

[1] A package containing our data is available at https://doi.org/10.5281/zenodo.7091926.

Table 1
Comparison of our study to other related work, based on the research field, methodologies, and analyses performed.

Topic	This study	Garousi and Mäntylä (2016)	Garousi and Fernandes (2017)	Karanatsiou et al. (2019), Wong et al. (2008, 2009) and Wong et al. (2011)	Garousi and Varma (2010) and Garousi (2015)	Farhoodi et al. (2013)	de Freitas and de Souza (2011)	Harrold (2000)	Bertolino (2007)	Orso and Rothermel (2014)
Field	Testing	All SE	All SE	All SE	All SE	Sci. SW	SBSE	Testing	Testing	Testing
Method	Quan., Qual.	Quan.	Quan.	Quan.	Quan.	Quan., Qual.	Quan.	Qual.	Qual.	Qual.
Research topics	✔	✔	✖	✖	✖	✔	✖	✖	✔	✔
Topic connections	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Keyword clustering	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Keyword connections	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Popular keywords	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Emerging topics	✔	✖	✖	✖	✖	✖	✖	✔	✔	✔
Declining topics	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Underexplored Con.s	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Potential connections	✔	✖	✖	✖	✖	✖	✖	✖	✖	✖
Popular papers	✖	✖	✖	✖	✖	✖	✔	✖	✖	✖
Top authors	✖	✖	✖	✔	✔	✔	✔	✖	✖	✖
Author location	✖	✔	✔	✔	✔	✔	✖	✖	✖	✖
Pub. Venue	✖	✖	✔	✖	✖	✔	✔	✔	✖	✖

Garousi et al. also performed bibliometric analysis, specifically, on the SE research communities in Canada (Garousi and Varma, 2010) and Turkey (Garousi, 2015). These studies differ from our own in their focus on the authors of publications, rather than research topics.

Farhoodi et al. reviewed literature related to scientific software, finding that many SE techniques are being applied in the field and that there is still a need to explore the usefulness of specific techniques in this context (Farhoodi et al., 2013). Their focus differs in both the analysis techniques, and in their focus on a specific software domain. In Section 4.4, we do observe the emergence of testing research related to scientific software.

De Freitas and de Souza performed a bibliometric analysis on the first ten years of research in search-based software engineering—the use of optimization techniques to automate tasks (de Freitas and de Souza, 2011). They identified the most cited papers, most prolific authors, and analyzed the distribution of the SBSE publications among conference proceedings, journals, and books. They described networks of collaborations and distributions of publications in various venues and identified the distribution of the number of works published by authors. Their study differs from ours in its focus on a particular research domain, as well as its focus on authors and venues over research topics.

2.3. Other related work

Purely qualitative analyses of testing research have also been performed. In Table 1, we contrast our study to those discussed below. None of these studies perform a full summarization or mapping of the testing field. Instead, they point out research areas that are emerging or that have had a major impact. The topics they discuss tend to form a subset of those in our characterization of the field. In addition, our quantitative analysis methods enable elaborate analyses of the field and the connections between topics not explored in these studies.

Harrold, in 2000, examined past research to identify areas of focus for future research (Harrold, 2000). These areas include improvements in integration testing, use of pre-code artifacts (e.g., specifications) to plan and implement testing activities, development of tools for estimating, predicting, and performing testing on evolving systems, and process improvements. Many of these predictions are now established topics in our map, such as black box testing, evolution and maintenance, and processes and risk.

Bertolino provided a summary of testing research in 2007, and identified achievements in the testing process, reliability testing, protocol testing, test criteria, object-oriented testing, and component-based testing as major advances (Bertolino, 2007). She identified outstanding challenges related to testing education, testing patterns, cost of testing, controlled evolution, leveraging users, test input and oracle generation, model-based testing, and testing of specialized domains, among others. Many of her achievements and challenges appear in our map as either keywords or full research topics.

Orso and Rothermel assessed research performed in the field between 2000–2014, asking colleagues what they believed were the most significant contributions and the greatest challenges and opportunities (Orso and Rothermel, 2014). The research contributions were categorized into the areas of automated test generation, testing strategies, regression testing, and support for empirical publications. The first three of those areas reflect research topics in our map. Challenges identified included better testing of modern, real-world systems, generation of test oracles, analysis of probabilistic programs, testing non-functional properties (e.g., performance), testing of specialized domains (e.g., mobile), and leveraging of the cloud and crowd. Some of these challenges – e.g., mobile and performance testing – are now research topics in our map.

3. Methodology

Software testing is one of the most popular and fast-growing areas of software engineering research (Orso and Rothermel, 2014). Although there are many surveys, mapping studies, and systematic literature reviews on individual topics, there is a lack of quantitative examination of the field as a whole—mapping research topics and their connections.

Our primary goal is to provide and analyze a "map" of the field of software testing, based on the many distinct research keywords that form the field and the connections between these keywords, linked through research publications. Our mapping is based on a quantitative method, co-word analysis, that places co-occurring phrases – in our case, author-supplied keywords – in a network. Within this network, keywords appear as nodes, with weighted edges indicating how often keywords are linked in publications. Sets of strongly co-occurring keywords form distinct clusters. This network structure offers a quantitative method to characterize the research field, which can be used as the basis of both qualitative and quantitative analyses.

Using this map, we examine how keywords are linked into *clusters*, characterize clusters using high-level research *topics*, examine the connections between keywords *within* and *across* clusters, and examine how interest in particular keywords and topics have changed *over time*. Specifically, we address the following research questions:

RQ1: What are the most popular individual keywords in software testing, as indicated by the number of publications or citations?
RQ2: What topics characterize the keywords connected *within* each cluster in the map?

RQ3: How are keywords and research topics most strongly linked *across* clusters?
RQ4: What keywords, topics, and connections have emerged or grown in popularity over the past five years?
RQ5: Which keywords and topics have shown the greatest decline in interest?

We begin, in **RQ1**, by examining the individual keywords targeted by authors. We are interested in identifying which keywords have been selected most often, and which receive the most citations per publication on average. We then move into analyses and characterization of the *connections* between keywords.

The goal of **RQ2** is to summarize each cluster. Keywords *within* a single cluster are highly interconnected, providing a basis for identifying *research topics* that encapsulate connected keywords. A topic as a keyword or phrase that connects multiple keywords. For example, "automated test generation" is not just a single keyword, but also a topic that connects other keywords such as "ant colony optimization" and "genetic algorithm" within the same cluster.[2] **RQ3**, then, focuses on the connections *across* clusters, and characterizes how keywords and research topics connect.

RQ4 and **RQ5** focus on an additional dimension, the average age of publications associated with each keyword. In **RQ4**, we identify keywords, topics, and connections between keywords and topics that have emerged or grown in popularity in the past five years. In **RQ5**, we examine keywords and topics with the oldest average date of publication—those with a potential decline in interest. These emerging and declining concepts offer insight into how the field is evolving.

To answer these questions, we (1) collected publications related to software testing (Section 3.1), (2) constructed a map, using co-word analysis, of clusters of connected keywords (Section 3.2), (3) removed unrelated or redundant topics (Section 3.3), and (4), analyzed the map and underlying data (Section 3.4).

3.1. Data collection

To gain an inclusive overview of software testing, we gathered publications from the Scopus database. Scopus is a comprehensive meta-database, covering many conference and journal publication venues. We retrieved all publications returned for the search term *"software testing"* on September 26, 2020. Only publications published in English were used. This collection included 57,233 publications.

Following a manual cleaning stage (see Section 3.3), 49,802 publications were included, including 36,774 conference papers, 11,640 journal articles, and 1388 other articles. Fig. 1 gives an overview of the number of publications published per year. Our aim was to capture a representative sample of the field, not all possible articles on software testing. When we quote specific numbers of publications, these numbers should not be taken as absolutes, but as the approximate commonality of a topic.

For each study, we gathered the title, author data (names, affiliations, locations), keywords, publication date, venue metadata (e.g., publisher, venue, volume, page numbers), number of citations, DOI, link, and language.

3.2. Map construction

To map testing research, we used co-word analysis (Peters and van Raan, 1993). Co-word analysis is a natural language processing method that extracts important phrases from a textual dataset and identifies their relationships in a network based on the number of times that two terms co-occur together in all documents. This technique assumes that terms that co-occur more often are more strongly related to each other. As a result, all identified terms are classified into clusters using co-occurrence to measure term similarity and depict the extracted terms and their relationship in a two-dimensional visualization.

We used VOSviewer (Visualization of Similarities Viewer) to analyze the collected data. VOSviewer is a tool that creates maps based on network data (Van Eck and Waltman, 2010). These maps provide visualizations that allow researchers to explore items and relationships. There are various methods for establishing connections between items in these networks, including co-authorship, co-occurrence, citation, bibliographic coupling, and co-citation.

We tested title, abstracts, index keywords, and author-supplied keywords as the unit of analysis and found that author-supplied keywords are the most promising way to identify research topics and their connections.

In this analysis, we considered 20 as the minimum threshold of keyword occurrences. This threshold places a minimal barrier before a keyword is "important" enough to incorporate. Keywords appearing in fewer than 20 publications were omitted. This threshold was chosen after experimentation as a way to control the level of noise and difficulty of interpretation of the dataset and map, while still avoiding potential loss of interesting and emerging topics. We then iteratively removed keywords that were unrelated to software testing (e.g., publications that used software as part of classroom testing) and merged redundant keywords (e.g., "automated test generation" and "automated test case generation") – see Section 3.3 – leaving a final set of 406 keywords.

VOSviewer produces maps based on a co-occurrence matrix – a two-dimensional matrix where each column and row represents an item – a keyword, in our case–and each cell indicates the number of times two keywords co-occur. This map construction consists of three steps. In the first step, a similarity matrix is created from the co-occurrence matrix. A map is then formed by applying the VOS mapping technique to the similarity matrix. Finally, the map is translated, rotated, and reflected. We provide technical details on VOSviewer's algorithm in Appendix.

In VOSviewer, a map is visualized in three ways: The network visualization, the overlay visualization, and the density visualization (van Eck and Waltman, 2014). We have used the network and overlay visualizations in this study, as well as the raw underlying data.

The network visualization is the standard view, displaying clusters of related items, connected with edges based on their co-occurrence. Fig. 3 shows the full network visualization that is produced. In Fig. 2, we highlight a small portion to explain how to interpret the map data.

In this map, each node is a keyword. Fig. 2 focuses on the keyword "software reliability". All keywords with a sufficiently strong connection to the targeted keyword are highlighted, while unrelated keywords are made partially transparent. The size of a node is based on the number of occurrences of the keyword. In Fig. 2, software reliability is targeted in approximately twice as many publications as "optimization".

Keywords are organized into clusters according to the process described above. Individual keywords can be linked across different clusters. However, the keywords within a cluster tend to be very closely linked with several other keywords within the same cluster. The color of the node indicates its cluster. In Fig. 2, software reliability is marked in light blue, and other nodes with the same color belong to the same cluster (e.g., "software reliability growth model").

Keywords that co-occur in publications are illustrated with an undirected edge. The thickness of the edge indicates how many

[2] Both are algorithms often used to generate tests, linking all three keywords as part of the same topic.

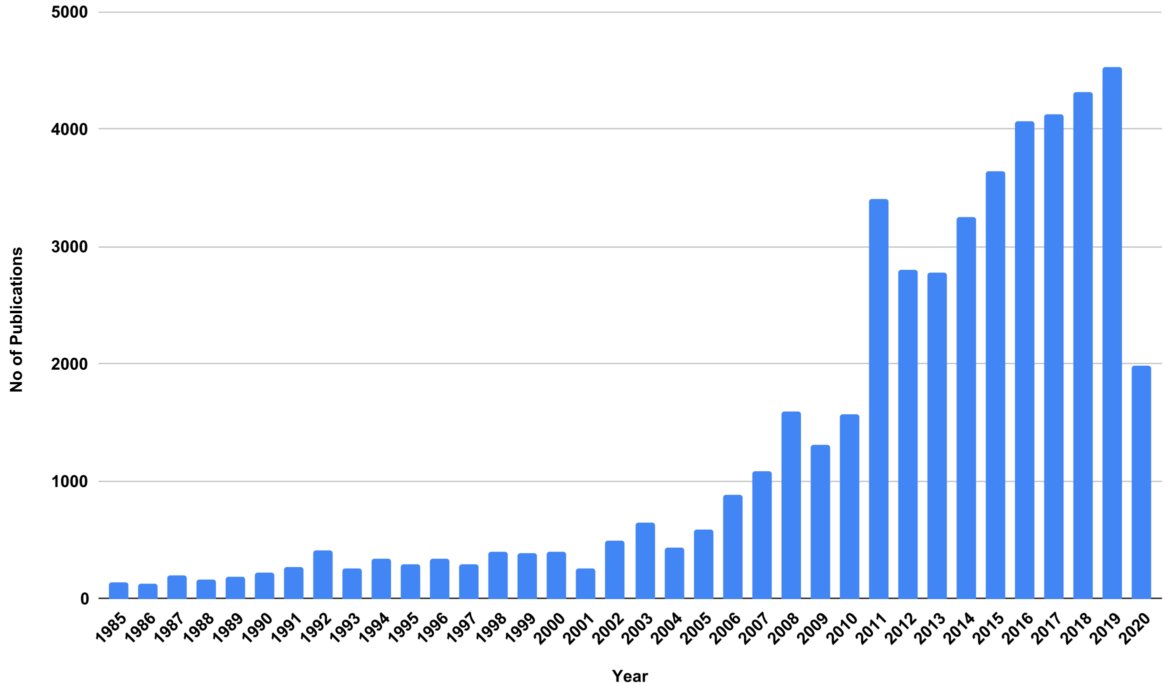

Fig. 1. Number of publications per year retrieved from Scopus.

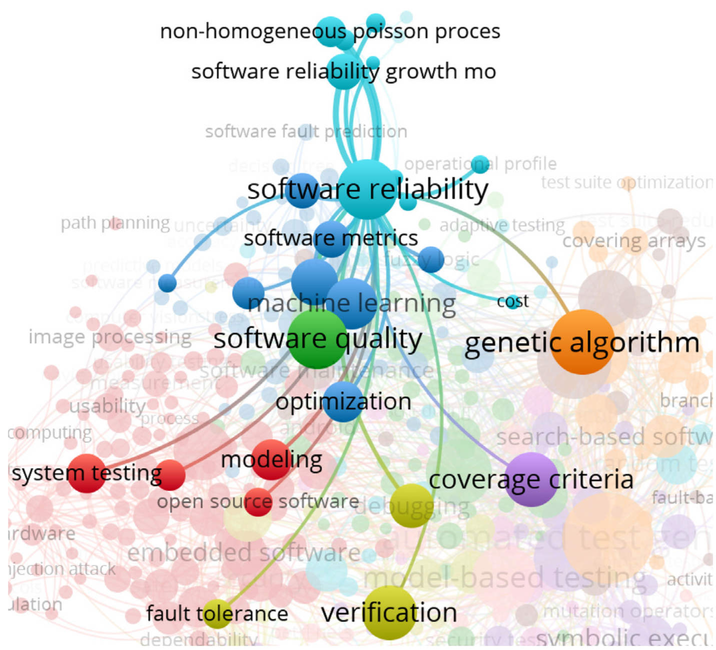

Fig. 2. Topics associated with software reliability. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

publications have targeted both keywords. In Fig. 2, software reliability and software reliability growth model share a stronger association (co-occurring in 35 publications) than software reliability with coverage criteria (5 publications). A user-controlled threshold determines the minimum connection strength for visible edges. We used the default, four publications, to control the level of noise when using the visualization for interpretation. When performing quantitative analyses, we consider all connections, regardless of strength.

The overlay visualization uses colors to indicate certain properties of a node, like the average number of citations that publications targeting a keyword have received, instead of using colors to show the cluster. In our case, we use this visualization to analyze the average age of publications targeting a keyword.

3.3. Data cleanup

The initial data included keywords that were either redundant or irrelevant:

- There are a small number of keywords unrelated to software testing, as the initial sample was gathered using a broad search string. For example, there were keywords related to

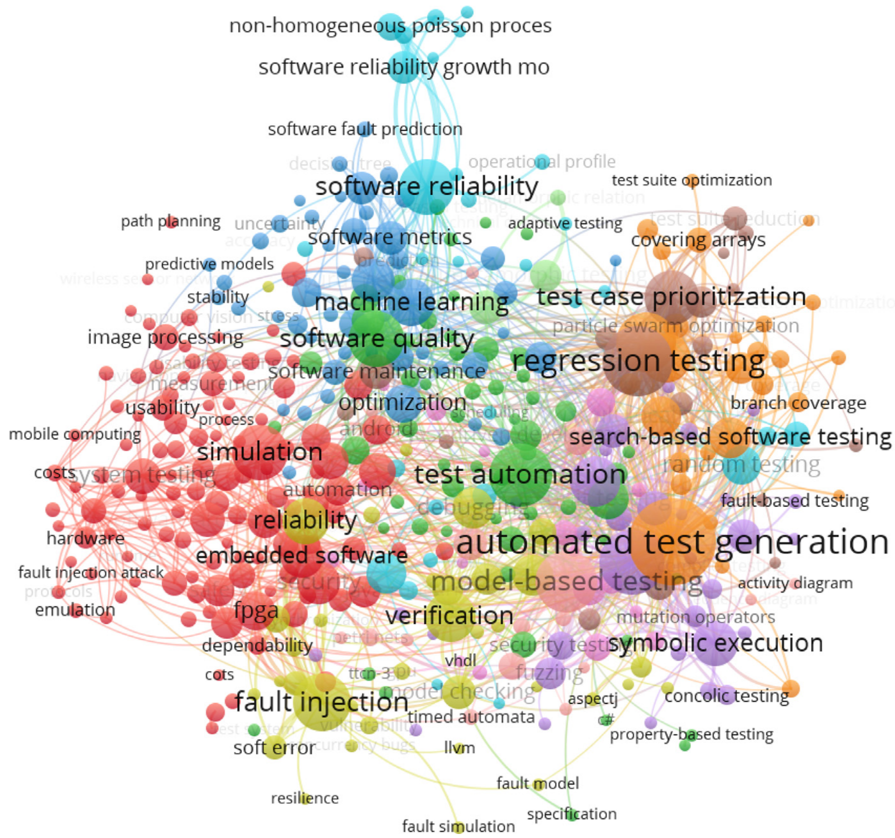

Fig. 3. A visualization of the connections between publication keywords.

software-based student examination or software-based testing of hardware. Additional keywords are either too generic to be considered as specific research concepts—e.g., "software testing"—or are research-related terms—e.g., "case study", "empirical study".

- Multiple keywords can refer to the same concept, and can be streamlined into a single keyword—e.g., "automated test generation" and "automated test case generation". The same keyword can appear in singular and plural form—e.g., "test case" and "test cases". There are also American and British English spellings (e.g., "prioritisation" and "prioritization").

To handle irrelevant and redundant keywords, we performed an iterative process. The authors discussed each keyword and came to a consensus. We removed irrelevant keywords from the map, as well as those considered too broad or generic. We removed publications targeting only those keywords, but retained publications that had additional keywords that remained in our set.

We merged redundant keywords. In performing this process, we limited merging to cases where a redundancy was obvious—primarily pluralization and British/American English. This was to limit the risk of biasing the underlying data that we are using to draw conclusions. We discussed each keyword and its alternatives, and came to a consensus on which keyword to use in all cases. We then replaced the merged keywords with the final keyword for each study and recreated the maps. We performed this process multiple times until we were satisfied that redundant keywords did not remain.

3.4. Data analyses

RQ1 (Popular Keywords): To identify the most common keywords, we sort the keywords by the number of publications that targeted that keyword, and examine those that are targeted in ≥0.50% of publications, or ≥250 publications. This threshold was chosen by examining the drop-off in significance over the ten most popular keywords and by considering the trade-off between clarity and giving a thorough impression of the testing field. A total of 20 keywords fall above this threshold (4.93% of keywords).

We also have examined which keywords have received the most citations per publication, on average. Here, we examine all keywords with an average number of citations ≥20. This threshold yields 23 keywords, and was chosen because it yields a similar quantity to the number of most common keywords, enabling clearer comparison.

RQ2 (Characterization of Clusters into Topics): We perform a qualitative characterization to summarize the field of software testing, supported by the clusters. We perform this summarization by assigning a small number of high-level "topics" to each cluster—keywords or phrases that connect multiple keywords. We have chosen these topics based on our interpretation of the keywords within each cluster. For example, "performance testing" is a keyword that connects, e.g., "load testing", "cloud computing", and "cloud testing".[3] Because that keyword summarizes many other keywords and connections within that cluster, "performance testing" can also serve as a topic that describes the cluster as a whole.

Some clusters can be summarized by a single topic, while others represent multiple topics. There is no case where *all* keywords are connected to *all* other keywords in the same cluster. Often, two keywords are only indirectly connected through other keywords—e.g., "random testing" is connected to "reliability" and

[3] "load testing" and "cloud testing" are forms of performance testing that target "cloud computing".

"adaptive random testing", but the latter two are not directly connected in our sample.

We often observed small sub-groupings of keywords within a topic. In such cases, we assign both a topic and a *subtopic*. For example, the topic of test creation guidance can be broken into four distinct types of creation guidance. A keyword can also belong to multiple topics or subtopics, linking topics within a cluster.

To assign topics, all authors examined the keywords within a cluster. We then grouped keywords into one or more groupings. To be grouped, keywords must have a direct or indirect (via a shared keyword) edge. This grouping was made based on our experience, literature, and the map data. We grouped keywords if they were used to perform the same activity, were a technology used to perform an activity, were a source of data for an activity, or had some other clear shared purpose. The proposed groupings were discussed until a consensus could be reached. We then identified either a keyword or concept that characterized each grouping. We discussed the options and reached a consensus on the topic to assign. In cases where a grouping could be split into smaller, but still distinct, groupings, subtopics were identified.

As all keywords must be grouped into a cluster, there are situations where a small number of individual keywords do not relate to the topics assigned to that cluster. We have tried to select topics that are as inclusive as possible.

RQ3 (Connections Across Clusters): In this question, we are interested in characterizing how keywords and topics are connected *across* clusters. To do so, we have examined two concepts—measurement of the *density of connections between clusters*, and the identification of *keywords that are connected to many other keywords*.

Connection Density: We can examine how clusters are connected by identifying the cases where the largest percentage of possible connections exist between keywords in two clusters, A and B:

$$\frac{(\text{Number of Connections Between Keywords in Clusters A and B})}{(\text{Number of Keywords in Cluster A}) * (\text{Number of Keywords in Cluster B})} \tag{1}$$

A measurement of 1.00 means that all keywords in Cluster A are connected to all keywords in Cluster B.[4]

To identify the most densely interconnected clusters, we measured the connection density between all pairs of clusters. We then focus on the connections with a density $\geq$0.12 (where at least 12% of all possible connections exist). This threshold was chosen after examination of the measurements—23 of the 45 pairs of clusters (50%) have a connection density above this threshold.

Connecting Keywords: Some keywords are singular research concepts, while others serve as "connecting keywords" that link many keywords together. We further characterize connections across clusters by identifying these connecting keywords.

To identify these keywords, we measure the number of keywords that each keyword co-occurs with outside of the cluster where that keyword is located. We analyze all keywords connected to $\geq$100 keywords in external clusters. This threshold was chosen as it yielded the same number of keywords as the threshold for the most popular keyword in RQ1 (20 keywords), enabling direct comparison.

[4] We employ the density because different clusters contain different numbers of keywords. This measurement offers a fairer basis of comparison than the raw number of connections between pairs of clusters.

RQ4 (Emerging Keywords, Topics, and Connections): In this question, we are interested in identifying emerging trends in testing research. We can do this by examining individual keywords, research topics, and connections between keywords.

We have classified any keywords with an average date of publication *later than June 2015* as our set of emerging keywords. This captures an approximate five year period ending with the date we took our sample of publications. A recent date implies one of two things about a keyword: (1) this is a new keyword, or (2), this is a older keyword that has received more attention in recent years.

We are also interested in examining the connections between keywords and topics, as related to the set of emerging keywords. There are a total of 2029 connections where at least one of the connected keywords is an emerging keyword, of which 1412 are cross-cluster connections and 617 are within-cluster connections. To focus our analysis, we focus on the cross-cluster connections, allowing us to also examine and characterize the emerging connections between topics.

To identify a subset of those 1412 connections for further exploration, we use the cross-cluster connection density to identify the pairs of clusters with the highest proportion of emerging connections. We have selected the ten pairs of clusters with the highest proportion of emerging connections for further examination, corresponding to a threshold of $\geq$3.5% of all connections between the two clusters consisting of emerging connections. For each pair of clusters, we group the connections by topic, then examine how the connection between these two topics is being shaped by the emerging connections between low-level keywords.

RQ5 (Declining Keywords and Topics): We address this question following a similar process to RQ4, based on *oldest* average dates of publication. 66 keywords met the threshold in RQ4. Therefore, we also examine the 66 keywords with the oldest dates of publication. This corresponds to a period from November 2001–May 2011.

These keywords represent concepts that are no longer receiving as much interest. This does not imply with certainty that such concepts are no longer relevant, or that they correspond to "solved" challenges. A keyword could represent (a) a topic or concept in decline (e.g., an older technology or approach that has been potentially superseded), (b) a well-established topic or concept with steady – but not growing – activity, or (c), a topic or concept that had a "boom" period in the past and a lower level of activity in recent years. Keywords may experience a resurgence. However, they have reduced relevancy to current development or testing trends, challenges, and research topics.

Before stating that a particular topic is in decline, we compare the list of keywords and topics with those in RQ4. We say that a topic is declining in interest if both (a) it has several keywords with older average publication dates, **and** (b), lacks keywords with recent average dates. By examining both the oldest and newest keywords, we can more carefully discuss whether a topic is potentially in decline.

4. Results and discussion

Our analyses are based on 406 keywords, which are mapped into 11 clusters. We analyze this map by identifying the most popular keywords by occurrences and citations (Section 4.1) and the overarching research topics of each cluster (Section 4.2), examining how keywords and topics are linked across clusters (Section 4.3), and exploring keywords, topics, and connections that are emerging or in potential decline (Sections 4.4–4.5). We also offer advice and further exploratory analyses in Section 5.

Table 2
Keywords targeted in at least 0.50% of publications (≥250 publications). Each keyword is named and described, and the number of publications where the keyword is targeted, percentage of the sample, average date of publication, and average number of citations per study are included.

Keyword	# Pubs.	Percent	Citations	Date	Description
Automated test generation	1068	2.14%	16.36	2013	The use of tools to generate full or partial test cases (Anand et al., 2013).
Regression testing	701	1.41%	14.03	2014	A practice where tests are re-executed when code changes to ensure that working code operates correctly (Korel and Al-Yami, 1998).
Mutation testing	596	1.20%	14.83	2014	A practice where synthetic faults are seeded into systems to assess the sensitivity of tests (Just, 2014).
Test automation	567	1.14%	9.71	2014	Tools and practices that enable automation of test execution (Fewster and Graham, 1999).
Model-based testing	552	1.11%	8.38	2014	Use of behavioral models to analyze the system, to design or generate test cases, or to judge results of testing (Anand et al., 2013).
Genetic algorithm	519	1.10%	10.10	2014	An optimization algorithm that models how populations evolve over time (Mitchell, 1998). Often used to automate tasks.
Fault injection	477	0.96%	6.12	2015	Injection of faults into a system for analysis (Voas, 1997).
Software quality	445	0.89%	8.14	2012	Means to define, measure, and assure the quality of software (Kitchenham and Pfleeger, 1996). Encompasses correctness and quality (e.g., performance or scalability).
Simulation	442	0.89%	4.53	2013	Simulated execution of a system. May encompass how to simulate (Herzner et al., 2007), testing in simulation (Mok and Stuart, 1996), or obtaining realistic results (Gay et al., 2017).
Software reliability	440	0.88%	12.71	2010	Means to define, measure, and assess the how quality changes over time (Crossley, 2000).
Test case prioritization	418	0.84%	17.42	2015	Automated techniques that select a subset of tests for execution (Rothermel et al., 2002).
Verification	366	0.73%	16.58	2012	Techniques that assess whether software possesses a property of interest, often using formal specifications (Pezze and Young, 2006). Testing is one verification technique.
Coverage criteria	362	0.73%	13.21	2012	Measurements used to assess the strength of a test suite based on how tests exercise code elements (Gay et al., 2015).
Combinatorial testing	349	0.70%	14.35	2015	A technique for generating or selecting test input, based on coverage of representative values (Nie and Leung, 2011).
Machine learning	326	0.65%	8.32	2017	Algorithms that make inferences from patterns detected in data. Used in, e.g., automation (Barr et al., 2015), predictive modeling (Turhan et al., 2009), or evaluation (Salahirad et al., 2019).
Reliability	306	0.62%	13.95	2013	Often a synonym for software quality, but can also refer to hardware quality or a blend of hardware/software.
Symbolic execution	295	0.59%	14.34	2014	Analyses where software is executed in an abstract form where one symbolic input matches many real inputs (Cadar et al., 2008).
Embedded software	268	0.54%	4.86	2014	Complex self-contained hardware and software systems (Zander et al., 2011).
Neural networks	266	0.53%	8.53	2015	Network structures inspired by the human brain, used in machine learning (Fukuda, 1992).
Security	265	0.52%	7.60	2015	Practices, tools, and techniques intended to prevent misuse of a system's capabilities or data (van Lamsweerde, 2007).

A visualization of the keyword map is shown in Fig. 3. An interactive version of this map can be accessed at https://greg4cr.github.io/other/2021-TestingTrends/topics.html or in the replication package.

4.1. RQ1: Popular keywords

We begin by identifying the most popular individual keywords, sorted by the number of publications (and percentage of the total sample). These keywords are listed in Table 2, along with a description, percentage of the sample, average age of publication (rounded to the year), and average number of citations per publication.

These keywords are those that the authors considered important enough to note as one of the core focuses of their work. There are certainly more than 326 publications in this sample that *use* machine learning, for example. However, the authors may not have listed machine learning as a keyword. Therefore, these keywords should be interpreted as the research concepts the authors felt were the most important and relevant.

RQ1 (Popular Keywords): The most common keywords tend to relate to automation, test creation and assessment guidance, assessment of system quality, and cyber–physical systems.

Automation offers promise for increasing the quality and efficiency of testing, and many keywords (e.g., automated test generation, test automation) relate to automation. Additionally, genetic algorithms and symbolic execution often enable automation. Test case prioritization enables efficient test execution, and regression testing is a process performed as part of a test execution pipeline. Combinatorial testing suggests an important subset

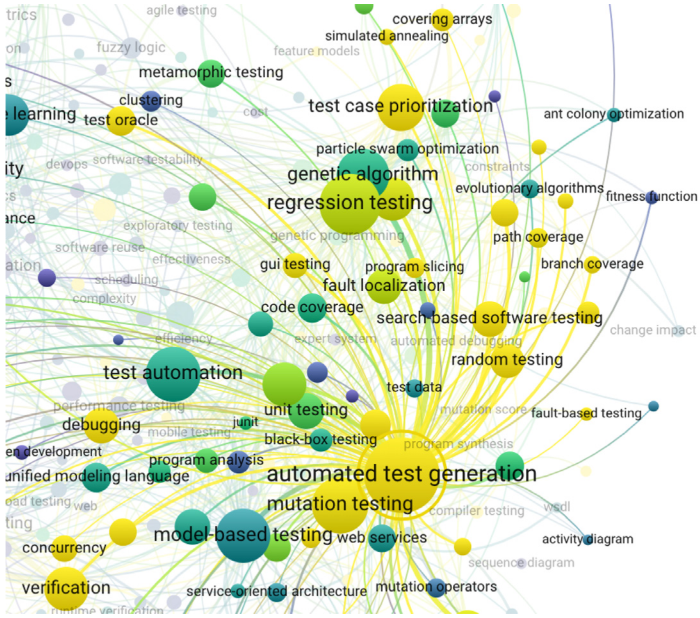

Fig. 4. A subset of keywords connected to automated test generation, colored by the average number of citations. Nodes in yellow attract a high number of citations ($\geq$14). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of test input to apply, often as part of automated generation. Models are often used to generate test input. Machine learning, including neural networks, supports prediction tasks related to automation.

Many of the remaining keywords relate to assessments of testing effectiveness or test creation guidance, including mutation testing, fault injection, and coverage criteria. Other keywords (software quality, reliability, verification, security) relate to the overall quality of the system, including its correctness, performance, and security. Finally, embedded software and simulation relate to systems combining software and hardware elements, which have high safety demands and unique testing activities (Gay et al., 2017).

The average number of publications per keyword is 75 (0.15%)—far below the number of publications targeting the top 20 keywords, indicating their importance. It is interesting that the most popular keyword is only a target of 2.14% of the sample, and that only five keywords are targets of over 1% of the sample. We believe this is an indication of the breadth of testing research. There are many challenges associated with testing, from test creation to automation, execution, assessment, and process. There are many ways to address each challenge, including algorithms and tools, human-driven activities, and studies of those working in the field. Even the median – 40 publications – is a reasonable body of work on any single concept.

We can compare the most-common keywords with the most-cited. In Table 3, we identify keywords that receive, on average, the most citations per publication.

> **RQ1 (Popular Keywords):** The most-cited keywords also relate to automation, test creation and assessment guidance, and assessment of system quality.

Some of these keywords are linked to the most common keywords—fault-based testing, test suite minimization, covering arrays, partition testing, evolutionary testing, and regression test selection, in particular. However, no keywords appear in both lists.

However, both the most common and most-cited keywords share common themes. Many of the keywords relate to automation (e.g., test suite minimization, random testing), test creation and assessment (e.g., testing strategies, data flow testing), or quality assessment (e.g., software fault prediction, sensitivity analysis). For example, Fig. 4 illustrates that many keywords associated with automated test generation receive a relatively high number of citations on average.

In general, the keywords in Table 3 are associated with a relatively small number of publications. They also have an older average date of publication, approximately 2010 versus 2014, allowing more time to attract citations. We hypothesize that these particular keywords (a) are related to themes that attract attention, and (b) are attached to a small set of publications containing a subset of particularly influential publications.

4.2. RQ2: Characterization of clusters into topics

By examining the connections between keywords, we can understand the context in which keywords form, grow, and thrive. Therefore, we have identified *research topics* characterizing the keyword clusters. These topics are detailed in Tables 4–5. We note a cluster ID assigned by VOSviewer, the number of keywords in that cluster, the density of connections between keywords within each cluster (the ratio of the number of existing within-cluster connections to the total possible within-cluster connections, $\binom{\text{Keywords}}{2}$), and the topics and subtopics assigned to that cluster. For each topic or subtopic, we list two example keywords that fall within that topic, and we briefly describe the meaning of the topic. For additional clarity, Fig. 5 outlines these topics, colored by the cluster they emerged from.

> **RQ2 (Characterization of Clusters into Topics):** Based on keyword clustering, testing research can be divided into 16 topics, with a further 18 subtopics.

While some of the topics within a cluster may seem independent, they are linked by connections between the underlying

Table 3
Keywords that received more than 20 citations on average per publication, with description, average number of citations, number of publications where the keyword is targeted, and average date of publication.

Keyword	Citations	# Pubs.	Date	Description
Testing strategies	55.50	26	2010	Guiding principles for test design and the testing process (Jamil et al., 2016).
Testing and debugging	48.48	21	2013	Debugging practices isolate and diagnose faults in the source code (Zeller and Hildebrandt, 2002). This keyword relates to the combination of testing and debugging techniques.
Partition testing	35.59	54	2005	Test input selection based on division of the system's input domain into partitions, based on a set of rules (Weyuker and Jeng, 1991).
Fault-based testing	34.48	33	2005	Use of pre-specified faults in a program to create and evaluate test suites (Morell, 1990). Mutation testing is an automated form of fault-based testing.
Constraints	33.76	25	2011	Conditions that must be met to accomplish a goal, e.g., for input to take a particular path in a program (Cadar et al., 2008).
Test suite minimization	32.97	39	2014	Process of reducing test suite size by eliminating redundant test cases (Yoo and Harman, 2012).
Random testing	32.88	218	2011	Testing software by generating random input (Arcuri and Briand, 2011).
Software fault prediction	31.82	38	2016	Prediction of fault-prone code using software metrics and fault metadata (Catal, 2011).
Covering arrays	28.50	96	2013	The set of test specifications selected during combinatorial testing (Nie and Leung, 2011).
Compiler testing	27.28	32	2014	Specialized testing practices for compilers, e.g., the selection of input programs to ensure the compiler conforms to its target language's semantics and syntax (Chen et al., 2020).
Object oriented modeling	25.75	20	2004	Model formats based on object-oriented design and object interaction (Rumbaugh et al., 1998)
Evolutionary testing	23.93	46	2011	The use of evolutionary algorithms (e.g., genetic algorithms) to generate test input or automate other tasks (Anand et al., 2013).
Test design	23.42	33	2013	The process of defining test cases (Copeland, 2004).
Regression test selection	23.40	58	2014	Practices to test cases for use during regression testing (e.g., only execute tests for changed code) (Rothermel and Harrold, 1996).
Monte Carlo	22.78	23	2015	A family of algorithms used for optimization, numeric integration, and probability assessment (Ahmed et al., 2020).
Alloy	22.18	22	2014	Language for expressing complex behavior and constraints in software (Jackson, 2019).
Automated debugging	21.76	38	2012	Automated debugging techniques (Zeller, 2001).
Adaptive random testing	21.51	95	2012	Random testing techniques designed to ensure input is evenly spread over the input domain (Chen et al., 2010).
Data flow testing	21.47	43	2011	Testing based on the flow of information between variable definitions and usages (Su et al., 2017).
Data flow	21.42	36	2008	Metrics for tracking the flow of information (Su et al., 2017)
Software standards	20.92	26	2009	Constraints, rules, and requirements that software or testing is expected to meet (RTCA/DO-178C, 2012).
Synchronization	20.07	29	2015	Practices for ensuring components are able to coordinate when completing tasks in parallel (Hong et al., 2012).
Sensitivity analysis	20.00	36	2012	Study of how uncertainty in system output can be traced to sources of uncertainty in its inputs (Douglas-Smith et al., 2020).

keywords. It is important, therefore, to examine both topics and keywords to come to a full understanding of a particular cluster. For example, random testing is a topic with widespread applicability. However, it is linked to Cluster 6 because random testing is often used to assess reliability or performance.

Within Cluster 2, the test automation topic encapsulates the emerging subtopic of mobile testing. Mobile testing is not as well-established as web application testing, but is clearly growing as a distinct research area. In the future, it may emerge as an independent research topic—perhaps even as its own cluster. Additionally, the model-driven development subtopic in Cluster 2 is related to – but also separate from – the model-based testing topic in Cluster 10. The latter focuses on technical aspects of modeling, while the former focuses on process and practices that may use these technologies. There are connections between the two, but they contain different keywords.

Cluster 1 is the least cohesive cluster. However, we can categorize many keywords under a core topic of system testing. In Cluster 2, there is a topic centered around test case types (e.g., unit testing). System testing is often grouped with these test case types. However, it is also a broader concept encompassing many different types of systems and system interfaces (e.g., embedded systems, operating systems, or databases). Several topics in our characterization also relate to system-level practices or domains, e.g., web, GUI, and performance testing. Those topics are established enough to stand independently, while the system testing topic in Cluster 1 acts as a broad umbrella.

4.3. RQ3: Connections across clusters

We analyze connections *across* clusters by measuring the connection density between pairs of clusters, and by identifying keywords that bridge clusters.

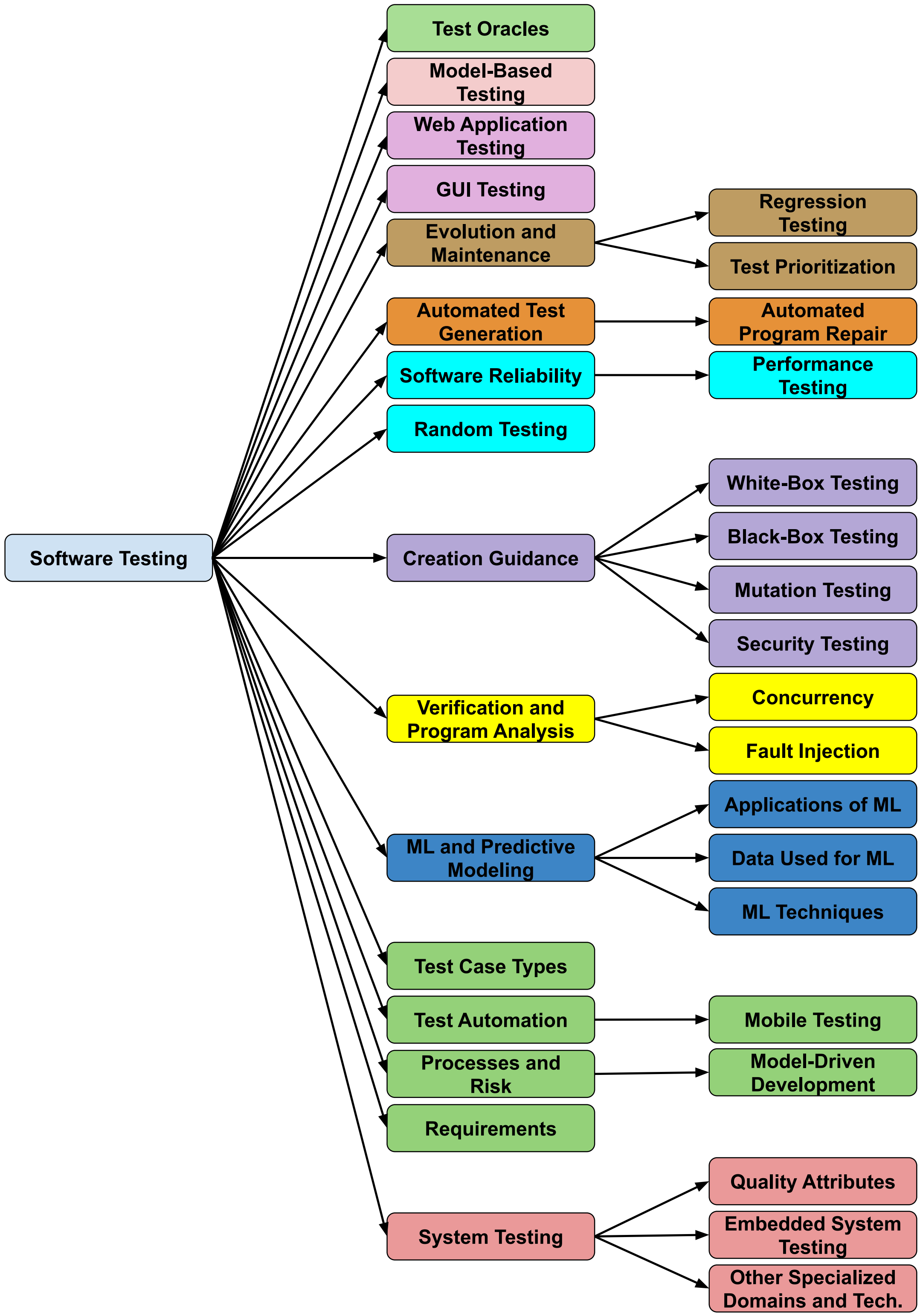

Fig. 5. Identified research topics (middle layer) and subtopics (final layer), colored by cluster. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Connection Density: Table 6 shows all cross-cluster densities, with those ≥12% are highlighted. 23 of the 45 pairs of clusters meet this threshold, indicating that many research topics are densely connected.

Table 4
An overview of clusters 4–11, including the cluster ID from VOSViewer, the number of keywords, inter-cluster connection density (percentage of possible connections between keywords), identified topics and subtopics, example keywords for each topic, and a brief description of each topic. Clusters are ordered from smallest to largest.

Cluster	Num keywords	Density	Topics and (Subtopics)	Example keywords	Description
11	4	100%	**Test oracles**	Test oracle, metamorphic relation	Test component that issues a verdict of correctness (Barr et al., 2015).
10	16	39%	**Model-based testing**	Model transformation, timed automata	Use of behavioral models to analyze the system, to design or generate test cases, or as oracles (Anand et al., 2013).
9	18	31%	**Web testing**	Web applications, javascript	Testing techniques, tools, and activities focused on verification of web-based applications (Bozkurt et al., 2010).
			GUI testing	Graphical user interface, finite state machine	Test design or generation techniques focused on exercising a system through its graphical interface (Qureshi and Nadeem, 2013).
8	19	47%	**Evolution and maintenance**	Program comprehension, change impact analysis	Practices for controlling and maintaining quality as the system changes over time (Murphy-Hill et al., 2012).
			(*Regression Testing*)	Regression testing, regression test selection	A practice where tests are re-executed when code changes to ensure that working code operates correctly (Korel and Al-Yami, 1998).
			(*Test Prioritization*)	Test case prioritization, test case selection	Automated techniques that select a subset of tests for execution (Rothermel et al., 2002).
7	28	48%	**Automated test generation**	Genetic algorithms, branch coverage	The use of tools to generate full or partial test cases (Anand et al., 2013).
			(*Automated Program Repair*)	Fault localization, genetic programming	Automated generation of patches for faulty programs (Martinez et al., 2017).
6	30	24%	**Reliability**	reliability growth, quality control	Means to define, measure, and assess the how quality changes over time (Crossley, 2000).
			(*Performance Testing*)	Load testing, cloud testing	Testing to assess performance and scalability of a system under different operating conditions (Helali Moghadam et al., 2019).
			Random testing	Adaptive random testing, statistical testing	Generation of random input for various purposes (e.g., assessing reliability or performance) (Anand et al., 2013)
5	32	30%	**Creation guidance**	Certification, test adequacy	Guidance for how a tester might approach test design—e.g., goals, input selection, and assessing test strength.
			(*White-Box Testing*)	Coverage criteria, data flow	Test creation based on source code (Gay et al., 2015).
			(*Black-Box Testing*)	Specification-based testing, black-box testing	Test creation based on requirements and other documentation (Whalen et al., 2006).
			(*Mutation Testing*)	Mutation score, mutation operators	Test creation based on synthetic faults seeded into a system (Just, 2014)
			(*Security Testing*)	Penetration testing, software vulnerability	Test creation to assess the ability of a system to prevent exploitation of vulnerabilities (Takanen et al., 2008).
4	35	31%	**Verification and analysis**	Dynamic analysis, static analysis	Analyses performed to ensure that software possesses properties of interest (e.g., correctness, resilience) (Pezze and Young, 2006).
			(*Concurrency*)	Parallelization, synchronization	Analyses of programs that execute over parallel threads or processes (Clarke et al., 1986).
			(*Fault Injection*)	Fault model, fault tolerance	Injection of faults into a system for analysis (Voas, 1997).

RQ3 (Connections Across Clusters): Clusters 5 (creation guidance), 7 (automated test generation), 8 (evolution and maintenance), and 11 (test oracles) are densely connected to several clusters. These clusters represent particularly multidisciplinary topics.

Cluster 8 appears in eight pairs, Clusters 7 and 11 appear in seven, and Cluster 5 appears in six. In particular, the pairings between Clusters 7 and 8 and between Clusters 5 and 11 have a higher connection density than the within-cluster densities of Clusters 1 and 2, indicating the dense interconnection between these topics.

As the most common keyword identified in our analysis, automated test generation has connections to keywords and topics in every other cluster. If a testing method exists, there will be an interest in generating tests for it (e.g., Clusters 5, 8, 9, and 10). Oracle creation often requires manual effort, leading to an interest in automated generation or reuse of oracles (Cluster 11). Further, machine learning offers the means to assist or enable automated test generation (Cluster 3).

Test oracles are a necessary component of almost all test cases, leading to dense connections with Clusters 5 (creation

Table 5
An overview of clusters 1–3, including the cluster ID from VOSViewer, the number of keywords, inter-cluster connection density (percentage of possible connections between keywords), identified topics and subtopics, example keywords for each topic, and a brief description of each topic. Clusters are ordered from smallest to largest.

Cluster	Num keywords	Density	Topics and *(Subtopics)*	Example keywords	Description
3	48	26%	**Machine learning**	Machine learning	Algorithms that make inferences from patterns detected in data (Turhan et al., 2009).
			(*Applications*)	Defect prediction, estimation	Applications of ML in software testing.
			(*Data Used*)	Metrics, complexity	Sources of data used to draw conclusions with ML.
			(*ML Techniques*)	Neural networks, deep learning	ML techniques used in testing research.
2	58	21%	**Test case types**	Unit testing, exploratory testing	Practices and levels of granularity for test design.
			Test automation	Test execution, testing tools	Tools and practices that enable automation of test execution (Fewster and Graham, 1999).
			(*Mobile Testing*)	Mobile testing, android testing	Testing techniques, tools, and activities focused on verification of mobile applications (Alshahwan et al., 2018).
			Processes and risk	Software quality, test-driven development	The organization, management, and testing process of a development team (Pezze and Young, 2006).
			(*Model-Driven Development*)	Model-driven development, model-driven testing	Development process based on use of models for analysis, code generation, and testing (France and Rumpe, 2007)
			Requirements engineering	Requirements engineering, traceability	Requirements indicate correct behavior. Verification often assesses conformance of code to requirements (Pezze and Young, 2006).
1	118	20%	**System testing**	System testing, user interfaces	Test cases that interact with an external system interface (Pezze and Young, 2006).
			(*Quality Attributes*)	Usability, software performance	Non-functional properties of a system assessed as part of quality assurance (Bass et al., 1998)
			(*Embedded Systems*)	Real-time system, simulation	Complex self-contained hardware and software systems (Zander et al., 2011).
			(*Other Specialized Domains*)	Open source software, image processing, autonomous vehicles	System types (e.g., databases, virtual reality, operating systems) or technologies (e.g., XML, Java) with dedicated testing approaches.

Table 6
Connection density between pairs of clusters. Cross-cluster densities ≥0.12 are highlighted. Densities in italics represent within-cluster densities for each cluster.

Cluster	1	2	3	4	5	6	7	8	9	10	11
1	*0.20*	0.08	0.09	0.11	0.06	0.06	0.06	0.07	0.07	0.09	0.07
2		*0.21*	0.08	0.09	0.10	0.08	0.09	**0.13**	0.10	**0.14**	0.11
3			*0.26*	0.10	0.08	0.08	**0.12**	**0.13**	0.05	0.07	**0.15**
4				*0.31*	**0.15**	0.08	**0.12**	**0.12**	0.07	0.10	**0.12**
5					*0.30*	0.08	**0.20**	**0.15**	**0.13**	**0.14**	**0.21**
6						*0.24*	0.10	0.11	0.06	0.06	**0.12**
7							*0.48*	**0.23**	**0.14**	**0.14**	**0.15**
8								*0.47*	**0.13**	**0.15**	**0.20**
9									*0.31*	0.10	**0.17**
10										*0.39*	0.09
11											*1.00*

guidance), 6 (reliability, performance, random testing), 8 (regression testing), and 9 (web and GUI testing). In addition to the above-mentioned connection to automated generation, machine learning offers a means to automate the creation of oracles (Cluster 3). Oracles are also a natural part of verification and different program analyses (Cluster 4).

Maintenance has implications on multiple aspects of testing, such as costs and quality. Maintenance needs affect the tasks performed during test automation (Cluster 2). Test prioritization also uses the same information that guides test creation to select tests (Cluster 5), and can be assisted using machine learning (Cluster 3). Both regression testing and test prioritization are performed for GUIs and web applications (Cluster 9), and can make use of models (Cluster 10). Further, analyses related to program and test evolution are often connected to other analyses in Cluster 4.

Test creation practices (Cluster 5) also connect broadly. Beyond automated test generation, test oracles, and test prioritization, several creation practices either have adaptations for model-based testing (Cluster 10) or for web and GUI testing (Cluster 9). In addition, there are connections between verification and test creation practices (Cluster 4)—e.g., black box testing and verification are connected through specifications, and security testing and analysis are related.

Clusters 1, 2, 3, and 6 have the least-dense connections to other clusters. Clusters 1 and 2 are both large clusters with multiple topics and subtopics that are distinct, but closely-related. Connections exist to other clusters, but may be less common, as these two clusters already represent a broad set of keywords. Reliability and performance testing (Cluster 6) and various forms of predictive modeling in Cluster 3 are also often pursued as standalone topics, but can be connected to other topics. Out of all density measurements, the lowest was between Cluster 3 (machine learning) and Cluster 9 (web and GUI testing), with 5% of possible connections existing in publications.

Connecting Keywords: In Table 7, we list all keywords that are connected to at least 100 keywords in external clusters.

Table 7
Keywords that are connected to at ≥100 keywords in clusters other than the one where the keyword is assigned (both keywords are targeted in at least one study). Each keyword is named and described, and the number of connected keywords (in external clusters, and in total) are listed.

Keyword	Connected (External)	Connected (All)	Position in Table 1	Description
Automated test generation	217	243	1	See Table 2.
Software quality	164	202	8	See Table 2.
Mutation testing	164	184	3	See Table 2.
Regression testing	157	174	2	See Table 2.
Test automation	152	200	4	See Table 2.
Model-based testing	150	163	5	See Table 2.
Coverage criteria	147	168	13	See Table 2.
Verification	143	164	12	See Table 2.
Genetic algorithm	139	161	6	See Table 2.
Machine learning	132	161	15	See Table 2.
Test case prioritization	119	134	11	See Table 2.
Software maintenance	119	130	–	Practices for controlling and maintaining quality as the system changes over time (Murphy-Hill et al., 2012).
Debugging	117	135	–	See Table 3.
Unit testing	114	136	–	A practice where tests are created for a small, isolated unit of code (typically a class) (Pezze and Young, 2006).
Software reliability	114	132	10	See Table 2.
Reliability	108	125	16	See Table 2.
Fault injection	106	128	7	See Table 2.
Static analysis	101	120	–	Analyses performed without executing the code (e.g., inspection or symbolic execution) (Pezze and Young, 2006).
Mutation analysis	101	116	–	Analyses of programs or tests performed using injected mutations (Just et al., 2011).
Unified modeling language	101	111	–	A family of techniques for modeling and analyzing program behavior (Rumbaugh et al., 1998).

RQ3 (Connections Across Clusters): Twenty keywords serve as "connectors" between clusters, reflecting multidisciplinary concepts (e.g., software quality), common test activities (e.g., unit testing), and common sources of information for test creation (e.g., coverage criteria).

For comparison, we also list the total number of connected keywords, and the position that the keyword had in Table 2 (if it appeared in the most commonly-targeted keywords). Many of the connecting keywords are also among the most common occurring keywords, with automated test generation on top of both lists. The exact positions of keywords shift in the ordering, but 14 of the 20 most common keywords are also connecting keywords. The most common keywords tended to relate to automation, test creation and assessment guidance, assessment of system quality, and cyber–physical systems. These concepts – especially the first three – are broad, with wide-ranging applicability. That suggests that popularity of a keyword is not only a reflection of a particular concept, but on its multidisciplinary applicability.

In contrast to Table 1, we see a notable rise in the position of software quality, coverage criteria, and machine learning. Software quality and machine learning are both very broad concepts, while coverage criteria are a common source of information and a target for testing, with applications in test creation guidance, automated test generation, quality assessment, prediction, and other areas.

We also see several keywords emerge: software maintenance, debugging, unit testing, static analysis, mutation analysis, and unified modeling language. These include broadly applicable concepts (maintenance, debugging, static analysis, mutation analysis), a common source of information (unified modeling language), and a common testing activity (unit testing).

Six of the most common keywords do not meet the threshold for connecting keywords—simulation (93 external connections), combinatorial testing (96), symbolic execution (83), embedded software (85), neural networks (83), and security (82). All six are multidisciplinary concepts, but are more specific – rather than broad – concepts (combinatorial testing, symbolic execution, embedded software, neural networks).

4.4. RQ4: Emerging keywords, topics, and connections

A visualization of the map of keywords, colored by average year of publication, is shown in Fig. 6. Yellow nodes have an average date of 2016 or newer. Blue nodes have an average date of 2010 or earlier. A gradient between blue and yellow represents 2010–2016. We examine keywords, topics, and connections that have emerged or grown in interest since June 2015.

An interactive version of this map can be accessed as an overlay at https://greg4cr.github.io/other/2021-TestingTrends/topics.html by selecting "Avg. Pub. Year" under the "Color" option.

Keywords and topics: Sixty-six keywords (16.26%) represent new emerging concepts or have received significant recent attention. Fig. 7 links these keywords to their respective research topic. From these results, we can make several observations:

- Many of the growth areas map to shifts in technology. There is growing interest in web applications, relating to technologies (JavaScript), testing tools (Selenium), and testing techniques. There is a similar emergence of mobile applications, in both the subtopic of mobile testing in Cluster 2 (android testing, mobile testing) and technologies in Cluster 1 (mobile applications, smartphone).
- Machine learning has advanced many fields. Unsurprisingly, it is also one of the largest growth areas in testing. The

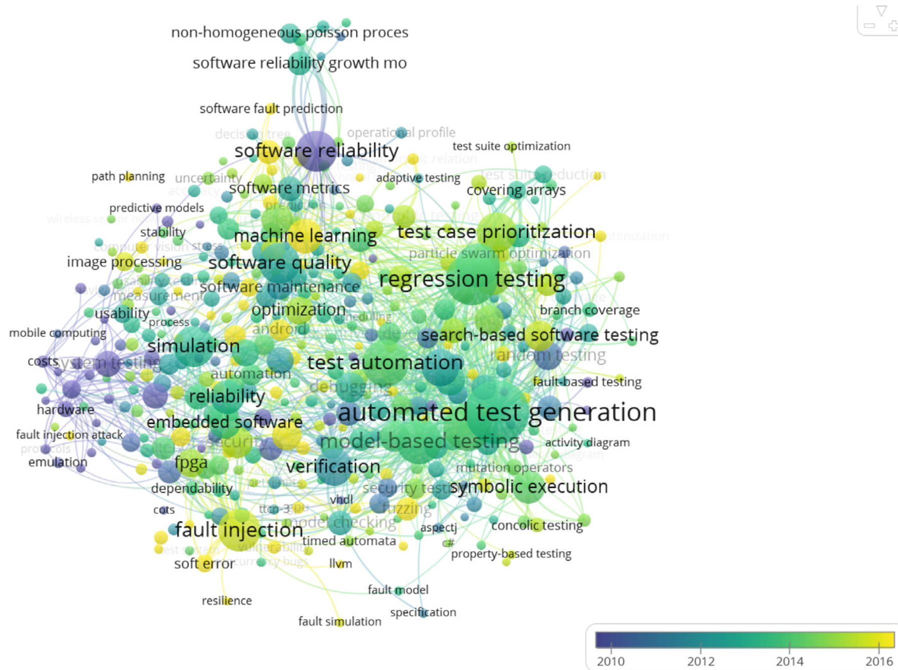

Fig. 6. The map of keywords, colored by the average year of publication. Note that "2010" should be read as $\leq$ 2010 and "2016" should be read as $\geq$2016. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

keyword "machine learning" has an average publication date of October 2016, and keywords have emerged related to applications, data, and specific techniques for ML. "Deep learning" is one of the newest keywords (average date of September 2018).

- Keywords have also emerged targeting ML and AI-based systems. From the embedded systems and "other domains" topics, we see keywords related to autonomous vehicles, computer vision, image processing, and augmented reality. All of these areas require specialized testing approaches. Autonomous vehicles, in particular, may grow into its own independent subtopic in the future.
- There is growing interest in energy consumption. This is connected to mobile applications, and a shift to portable devices that rely on batteries. This also reflects growing interest in sustainability and environmental impact of software.
- Automated program repair has emerged as a subtopic. The core keyword has one of the newest average publication dates (March 2017), and its connected keywords (e.g., program synthesis) also have recent dates.
- Fuzzing and search-based approaches (swarm intelligence, ant colony optimization) have emerged as test generation techniques. Fuzzing, notably, has seen application in general and security-focused testing topics. Security-related keywords are also active and growing.

> **RQ4 (Emerging Keywords, Topics, and Connections):** Emerging keywords and topics relate to, or incorporate, web and mobile applications, machine learning and AI – including autonomous vehicles – energy consumption, automated program repair, or fuzzing and search-based test generation.

Connections: We focus our examination on ten pairs of clusters with the highest proportion of emerging connections to the number of possible connections ($\geq$3.5%). The connected clusters, and their associated topics, have a rapidly evolving relationship.

- Cluster 11 (test oracles) with Clusters 5 (creation guidance; 8.59% of connections are emerging), 3 (machine learning; 6.77%), 8 (evolution and maintenance; 5.26%), 6 (reliability; 4.17%), 9 (web application and GUI testing; 4.17%), 2 (test case types, test automation, processes and risk, and model-driven development; 3.88%), and 4 (verification and program analysis; 3.57%).
- Cluster 7 (automated test generation) with Clusters 9 (4.36%) and 3 (3.57%).
- Cluster 5 with Cluster 9 (4.69%).

The highlighted connections between topics are shown in Fig. 8 for topics connected with test oracles, and in Fig. 9 for other topics. For each connection between topics, a small number of example connections between keywords are shown.

> **RQ4 (Emerging Keywords, Topics, and Connections):** Web applications and scientific computing require targeted testing approaches and practices, leading to emerging connections to many topics. Test oracles are also a rapidly-evolving topic with many emerging connections. Machine learning has emerging potential to support automation.

We make several observations about these emerging connections:

- Test oracles appear often because (a) Cluster 11 is a small cluster, (b) this topic has the largest percentage of emerging keywords, and (c), this topic is naturally connected to all other topics. Research interest in test oracles is growing (Barr et al., 2015; Fontes and Gay, 2021), and effective

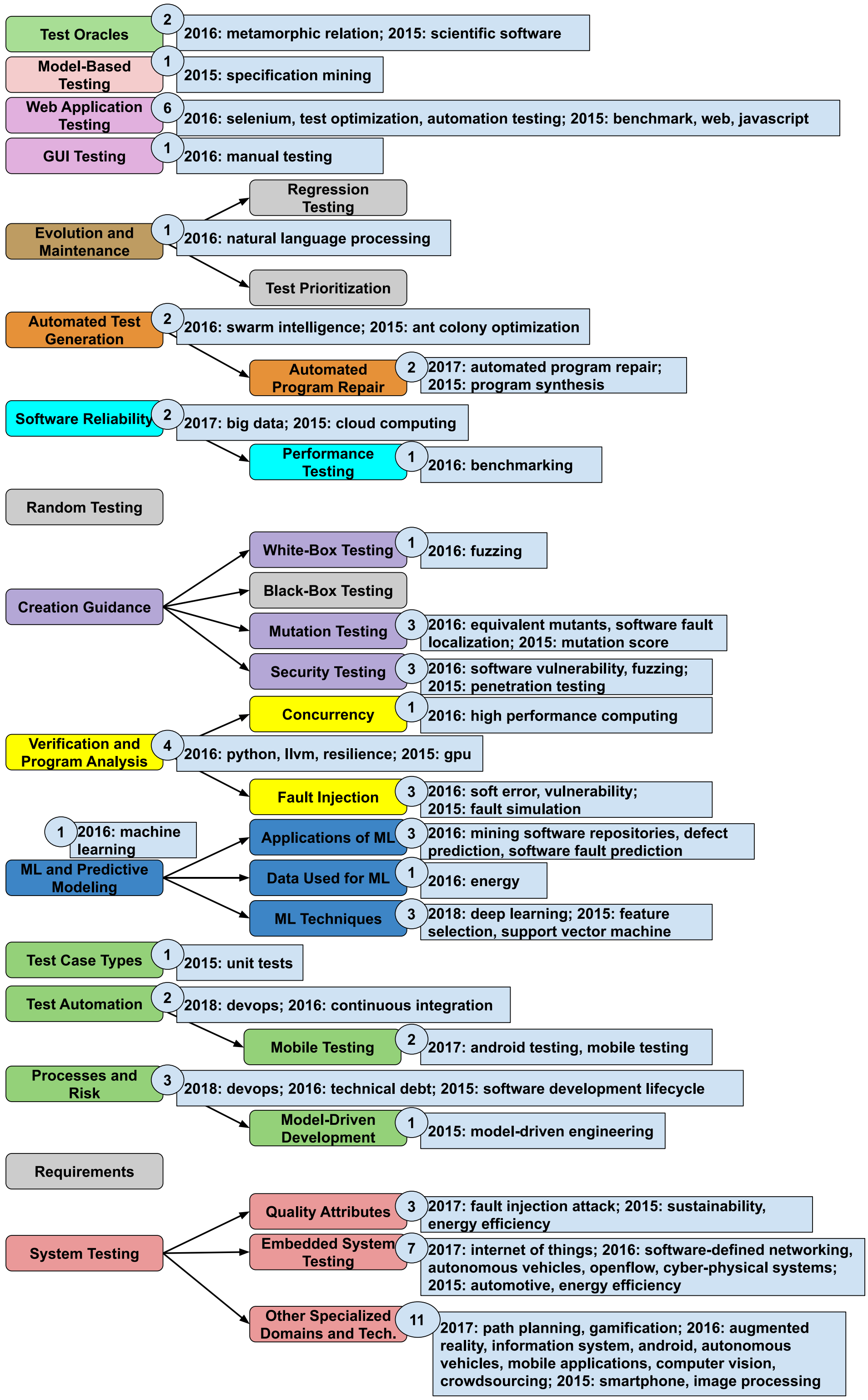

Fig. 7. Keywords with an average publication date **newer than June 2015**, along with their associated research topic. The number next to the list of keywords indicates the number of emerging keywords. Topics colored in gray are those without emerging keywords. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

oracles are needed for emerging domains such as web applications. The relationship between oracles and different testing practices is not well understood yet, leading to many emerging connections. Further, interest is growing in the use of machine learning to generate test oracles (Fontes and Gay, 2021).

- The keyword "scientific computing" is part of Cluster 11, due to its frequent connection with metamorphic testing. Inspection of the emerging connections makes it clear that software testing for scientific computing is emerging as a distinct domain of interest, with major connections to Cluster 2 and 5.

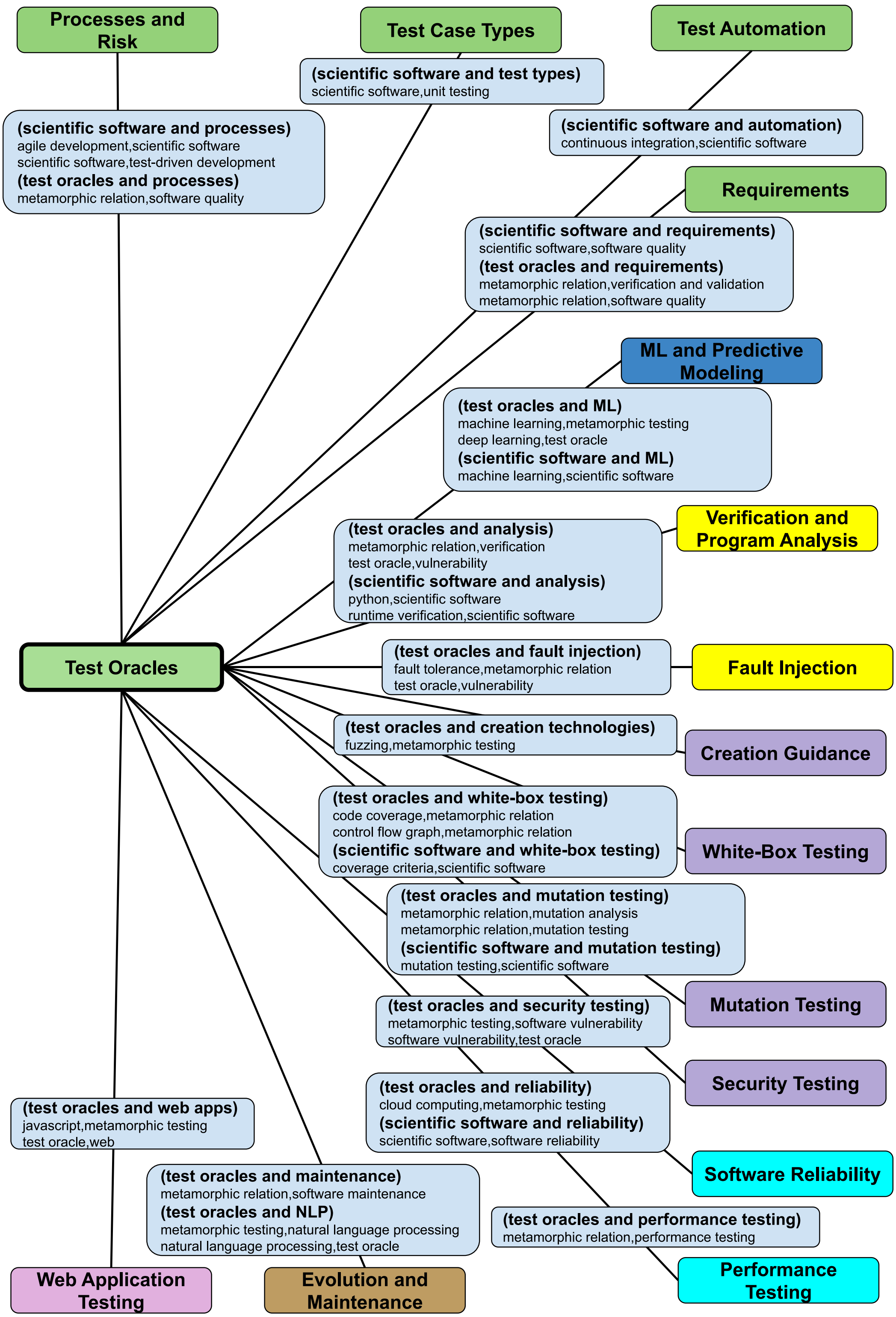

Fig. 8. Emerging connections, connected by research topic with test oracles, for the cluster pairings with highest ratio of emerging to total connections.

- As in many other areas of software development, machine learning offers the potential to automate tasks that traditionally require significant human effort, such as test and oracle generation and program repair.
- Test creation practices of many types (including white-box, black-box, mutation, and security) are emerging for web applications.
- New test generation approaches are emerging for GUIs and web applications.

4.5. RQ5: Declining keywords and topics

Fig. 10 shows the 66 keywords with the oldest average date, with their associated research topic. In particular, we highlight

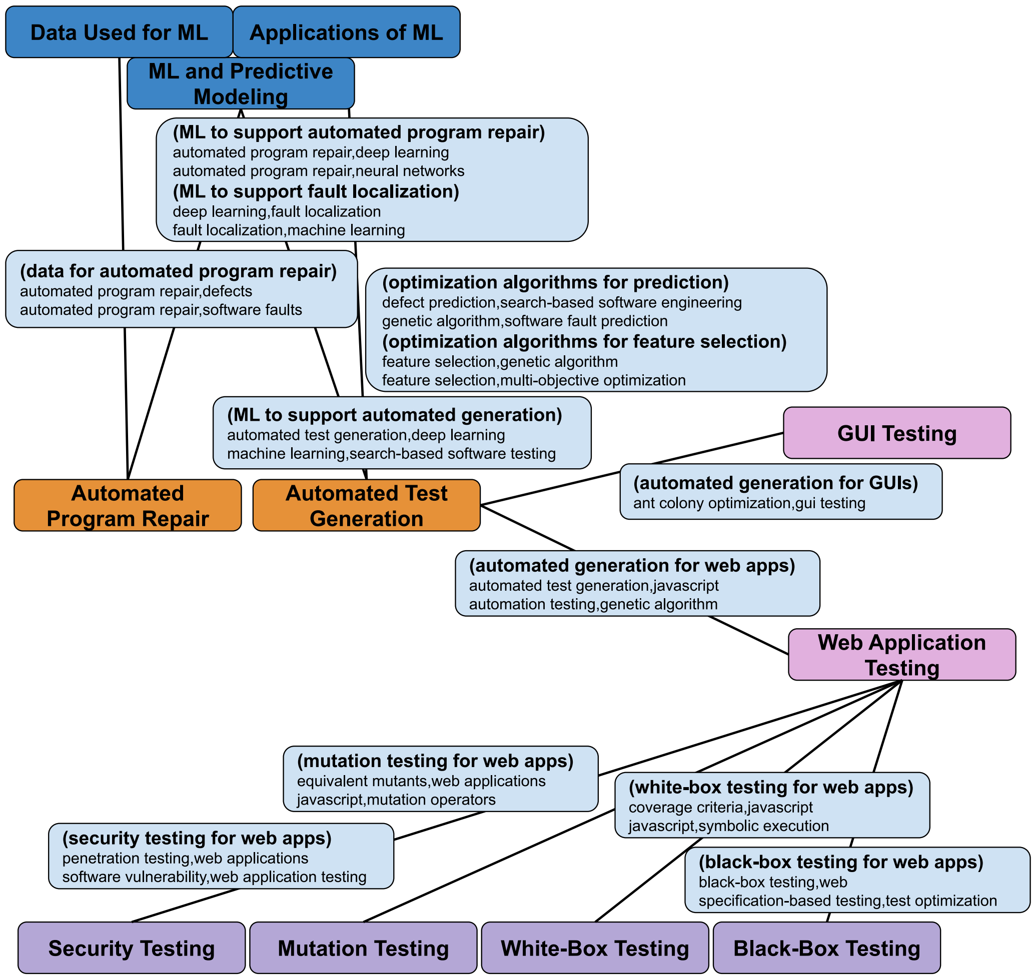

Fig. 9. Emerging connections, connected by research topic (excluding test oracles), for the cluster pairings with highest ratio of emerging to total connections.

three research topics or subtopics that we hypothesize may currently be in decline.

> **RQ5 (Declining Keywords and Topics):** Older average dates of publication and lack of emerging keywords suggest that keywords and topics related to random and requirements-based testing may be in decline.

Briefly, we examine these areas:

- Traditional random testing has been supplanted, to some extent, by semi-random approaches. As shown in Fig. 7, search-based and fuzzing techniques are growing in popularity. Both use sampling heuristics instead of applying pure random generation, retaining some of the benefits of random testing (e.g., scalability) while potentially yielding more effective results.
- Many of the keywords related to requirements and black-box testing have older average publication dates, indicating potential stagnation. Agile processes favor lightweight requirements (e.g., user stories) over formal and complex requirements. We hypothesize that this may have led to a shift in attention towards other sources of information for test creation.

We hesitate to state that these topics are "dying" or are solved challenges. However, we do see evidence that they have not seen notable growth in popularity or the emergence of new keywords in recent years. New application areas, techniques, or changes in development processes may lead to a resurgence in interest in the future.

5. Further analysis and advice to researchers

Both the high-level topic overview and the low-level map of connections between keywords can serve as inspiration for prospective and experienced researchers. We offer the following advice on how this data could inspire new research.

An overview of the testing field: For inexperienced researchers, the high-level topics offer an immediate "snapshot" that can be used to guide exploration of different research areas. The keywords illustrate key concepts that form research topics, and offer targeted suggestions on terms the researcher should examine in detail. Connections between those keywords illustrate how those concepts have been connected in practice, which may encourage critical reflection on both the individual concepts and how they relate. The emerging keywords and topics suggest areas that researchers may wish to pay attention to, and emerging connections clarify how these keywords fit into the field.

Understanding the context of a keyword or topic: Researchers can analyze the map to gain a data-driven view of the field for further planing and development. As a starting point, those interested in a keyword or topic can examine how that keyword or topic fits into the broader context of testing research.

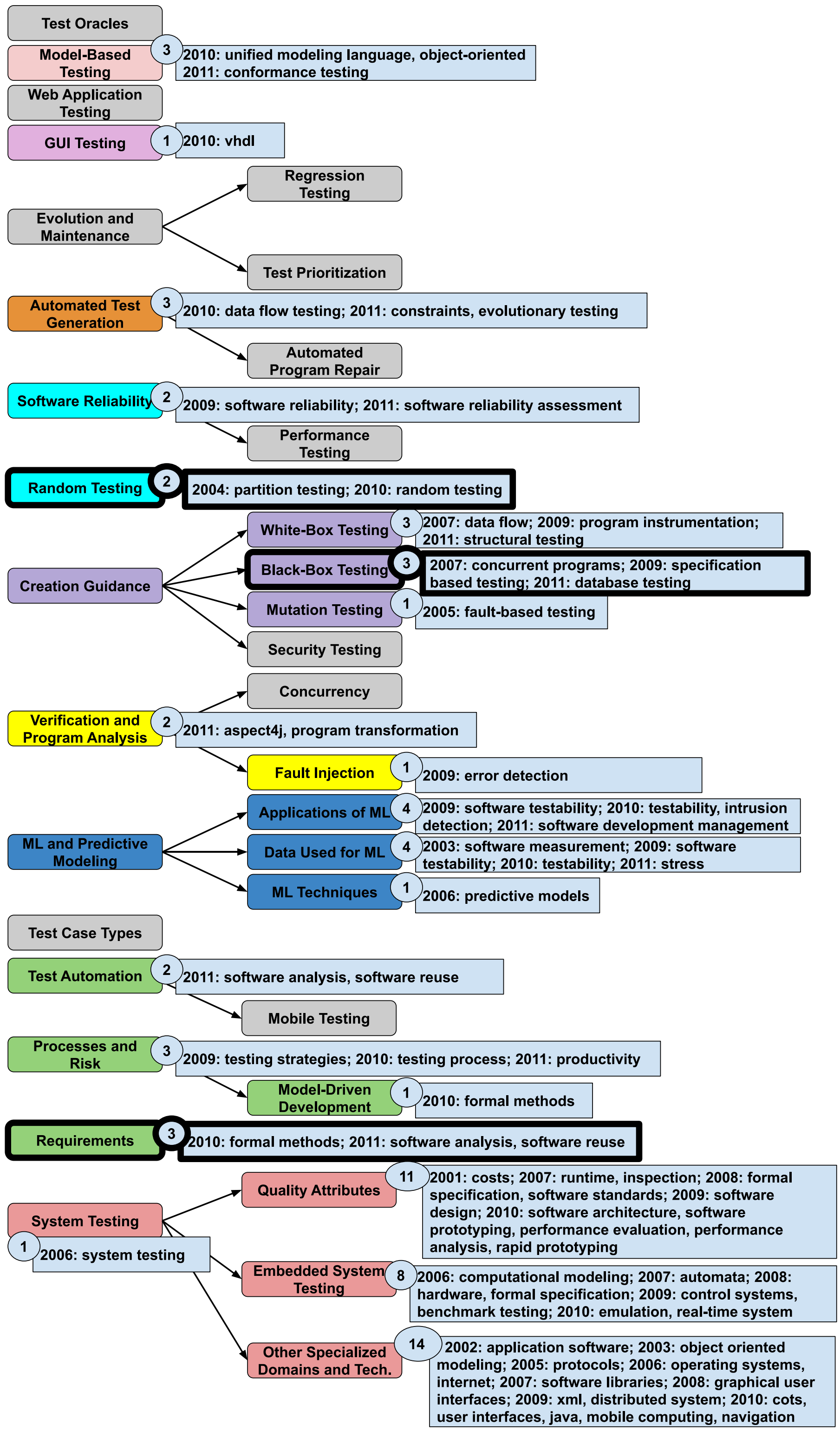

Fig. 10. Keywords with an average publication date **earlier than June 2011**, along with their associated research topic. Topics colored in gray are those without declining keywords. Topics with both declining keywords and a lack of emerging keywords are highlighted. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- What keywords are often associated with a keyword of interest? This may illustrate the type of research often conducted on this concept (or its associated topic), and natural areas of synergy between keywords or topics.
- Is interest in this keyword or topic growing, declining, or stable? The average date of publication may suggest the current level of interest (or lack thereof).

Identification of under-explored connections between keywords or topics: We hypothesize that the map data could potentially inspire future research through analyses of connections between keywords and topics. There are many ways connections could be analyzed. One is to identify keywords that have *under-explored* connections.

Specifically, an under-explored connection is one where (a) at least one publication has connected the keywords, but (b), the specific number of publications connecting those two keywords is relatively low—indicating potential for additional research exploration. Under-explored connections may serve as inspiration, suggesting concepts that could be connected in further research:

- Within a cluster, under-explored connections may suggest ways that concepts within a particular research topic could be more closely linked. For example, different mechanisms from automated test generation algorithms could be blended into hybrid algorithms.[5] An examination of under-explored connections could offer similar inspiration.
- Across clusters, we could identify either pairs of clusters or topics that could be more deeply connected in future research. In some cases, these may be topics that are already connected (e.g., automated test generation and white-box testing), but where there are opportunities for new research related to specific keywords or aspects of the topic (e.g., specific test generation algorithms).

There are different ways that under-explored connections could be identified and analyzed. As an initial exploration, first, one must identify a lower and upper bound on the number of publications linking keywords. As an example, in the network visualization, four publications are needed for an edge to be shown (by default). Therefore, one could adopt four publications as the threshold for this analysis and capture all connections in a short range of this threshold—e.g., 4–6 publications targeting a pair of keywords.

718 connections have a strength of 4–6 publications. To identify a subset for initial exploration, we can (a) focus on cross-cluster connections, and (b), use the cross-cluster connection density to identify the pairs of clusters with the most under-explored connections. Here, we specifically focus on the cluster pairings where ≥2% of all connections between the two clusters consist of under-explored connections. Six cluster pairings met the threshold: Cluster 11 (test oracles) with Cluster 9 (web and GUI testing, 2.63%), Cluster 5 (creation guidance, 2.34%), and Cluster 3 (machine learning, 2.08%), and Cluster 7 (automated test generation) with Cluster 8 (evolution and maintenance, 3.38%), Cluster 10 (model-based testing, 2.46%), and Cluster 5 (2.01%).

For these cluster pairings, we grouped the connections by research topic, then examined the meaning and potential application of the connections. In Fig. 11, we illustrate the identified connections, categorized by their associated research topics.

We make several observations about these connections. First, specific suggestions emerge for exploring connections in future research, including (among others):

- The relationship between mutations and test oracles.
- Use of mutation as part of automation program repair and test generation.
- Test generation based on specific modeling formats (e.g., object-oriented models such as activity diagrams).
- Reduction techniques for generated test suites and test cases.
- The relationship between test generation and program evolution (e.g., how often tests should be generated, how tests should be maintained).
- Generation of tests for regression testing.
- The use of specific optimization algorithms for test case prioritization.

In some cases, "under-explored" coincides with "emerging"—for example, test oracles with machine learning and web services. There are also cases where topics are well-connected in research (e.g., test generation and white-box testing) through different keywords (e.g., "coverage criteria" instead of "code coverage"). We retained keywords with minor differences in meaning, as even minor distinctions may be important. However, some connections may be well-explored under a different keyword. Even in such cases, there may be opportunity for further exploration related to these keyword differences, or connections based on concepts and technologies than have not been explored previously (e.g., specific generation algorithms or coverage criteria).

Identifying new connections between keywords: The *absence* of a connection between two keywords does not imply that the concepts cannot be connected. Consider keywords within a single cluster. Keywords lacking a direct connection may represent entirely incompatible concepts. However, in other cases, there may be a natural synergy between the two concepts that had not yet been considered. While the map cannot directly inform researchers which keywords *can* be connected, or how they can be connected, it can serve as a means to prompt brainstorming.

As an example, we can inspect keywords within a cluster that lack a direct connection to specific other keywords in their cluster. Cluster 8 (evolution and maintenance, with subtopics of regression testing and test prioritization) contains 19 keywords. There are 180 cases where two keywords lack a direct link within Cluster 8—e.g., "change impact analysis" and "test case reduction" are not directly connected in publications.

Not all of these cases offer obvious ideas for new research, but consideration of these cases may lead to inspiration. For example, we identified the following ideas:

- The use of change impact analysis as part of program comprehension, test case reduction, test suite minimization, or test suite reduction.
- The use of information retrieval and natural language processing to provide information for test case and suite reduction, selection, and minimization and for regression test selection.
- The use of regression test selection techniques for use as part of test case and suite reduction and test case selection.
- The use of program comprehension techniques for regression test selection.
- The relationship between evolution and maintenance of software with test case prioritization, minimization, and reduction.
- Service-oriented architecture and web services appear in this cluster because of close association with particular keywords (e.g., regression testing), but are only indirectly connected to the majority of the other keywords. The missing connections suggest the need for targeted test case prioritization, selection, reduction, and minimization approaches for service-oriented architectures and web services, as well as examination of the evolution and maintenance of service-oriented architectures and web services.

Similar ideas may emerge from inspecting missing connections within other clusters.

There are many ways that this map could potentially be analyzed beyond the simple exploration in this section. We suggest

5 As has been done for concolic execution and search-based test generation (Galeotti et al., 2013).

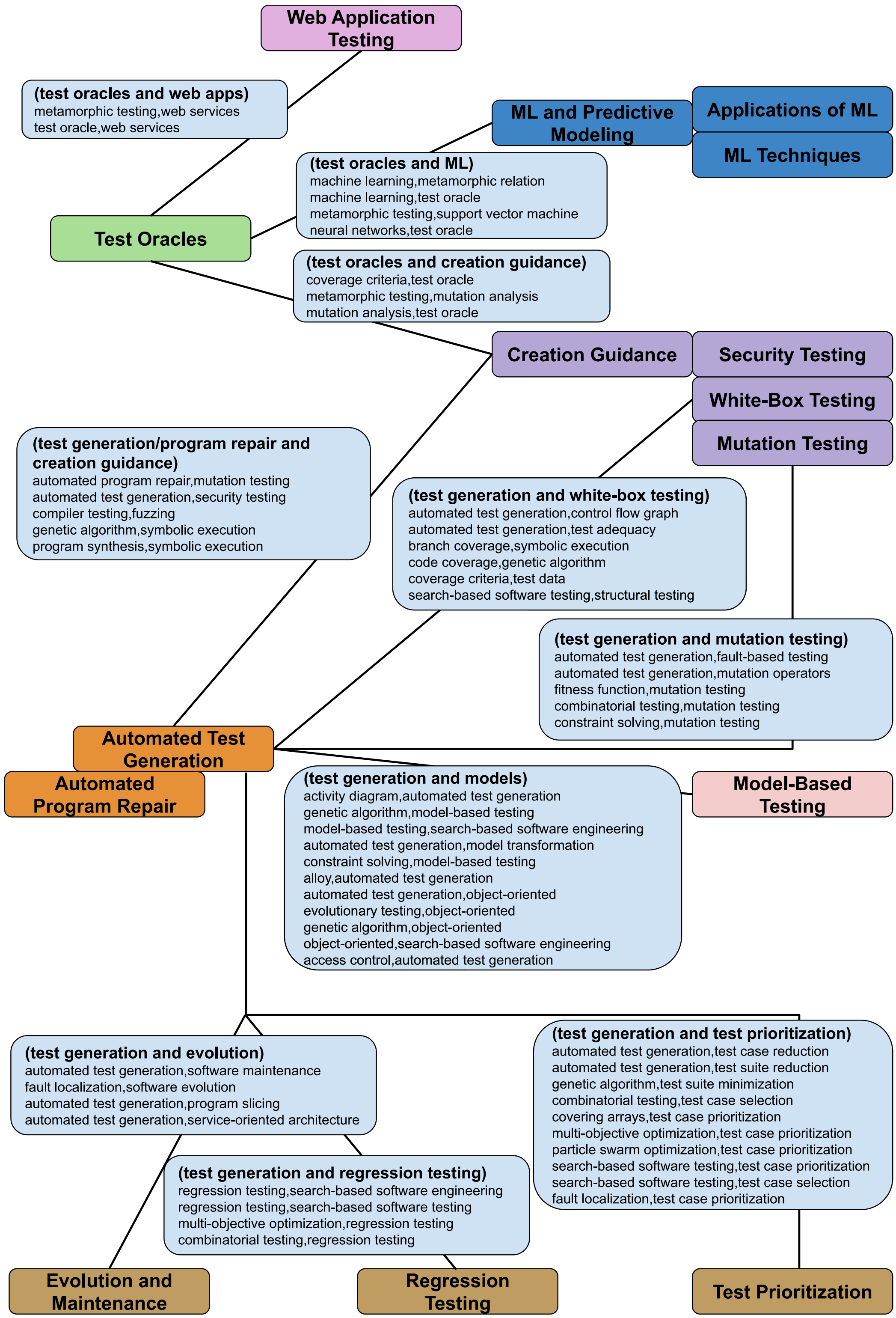

Fig. 11. Under-explored connections (keywords connected by 4–6 publications), connected by research topic, for the six cluster pairings with highest ratio of under-explored to total connections.

that researchers attempt to analyze different connection types, connection strength thresholds, and other aspects of the collected metadata (e.g., publication age or number of citations) in order to gain inspiration for new research or insight into the field.

6. Threats to validity

Conclusion Validity: VOSviewer was used to perform visualization. The design of this tool and the visualizations it produces could potentially bias the observations made. However, the tool is based on well-understood and established computational principles. Further, it has been used in over 500 bibliometric studies (e.g., Mohammadi and Karami (2020) and Mohammadi (2012)), in a large variety of fields and its assumptions have been verified by experts in these fields.[6] We have made efforts to verify the assumptions behind the analyses performed.

External Validity: Our study examined publications from the Scopus database, potentially omitting relevant venues for software testing research. Scopus is the one of the most comprehensive databases covering research publications (Thelwall and Sud, 2022), indexing content from 24,600 active conferences or journals and 5000 publishers.[7] Specifically, Scopus coverage for computer science research has been found to be better than other databases (Cavacini, 2015). Scopus also enables efficient export of the data we use to perform our mapping. Although some venues may not be indexed, many of the most important journals and conferences in the software testing field are included.

We used a single search string to build our sample. Other search strings (e.g., "software test") could have complemented the search process. However, our goal is not to capture all studies ever published in software testing. Rather, we require a sufficiently representative sample. We hypothesize that the additional value would be minimal compared to the filtering effort required. We believe that our sample of 57,233 publications is sufficiently large and representative to perform this analysis.

Internal Validity: We based our analysis on publications retrieved using the term "software testing". This pool of papers included publications unrelated to software testing, e.g., the use of software to test hardware or as part of student examination. We performed a manual process to remove unrelated keywords from the mapping. However, it is possible that some publications remain that are unrelated to the targeted research field. We believe that these are not enough to influence our observations.

Our analysis is based on author-supplied keywords, and not other sources of topic information, e.g., titles or abstracts. The use of keywords introduces a risk that publications are mislabeled (e.g., authors used the wrong term), or that important concepts are omitted. Still, author-supplied keywords are a clear and appropriate means to capture the structure of software testing research. Author-supplied keywords are regularly used in other bibliometric analyses (Mohammadi and Karami, 2020; Jamali et al., 2020; Liao et al., 2018) and have been found to effectively reflect structures in research fields (Liao et al., 2018; Li et al., 2016). Even if relevant keywords are omitted, the concepts the authors felt were most important are reflected. While there is potential inaccuracy, it is likely that the selected keywords are close to correct. Alternative methods carry similar risks. Automated or external categorization can also be inaccurate and potentially violates the intent of authors. Other sources of information, such as titles or abstracts, introduce noise and are difficult to use to categorize publications.

We applied a threshold of a minimum of 20 studies before a keyword appeared in our dataset or map. We used this threshold to omit minor or highly obscure keywords and to control the level of noise in the map. This risks also omitting emerging keywords. We tried lower and higher thresholds then we concluded that the current threshold is enough to cover terms with lower frequency and provide a meaningful and lower scatter network of the keywords. It should be noted when we tested lower and higher thresholds, the overall patterns did not change significantly.

7. Conclusion

Testing is the primary means of assessing software correctness and quality. Research in software testing is growing and rapidly-evolving. Based on the keywords assigned to publications, we seek to identify predominant research topics and understand how they are connected and have evolved.

We have applied co-word analysis to characterize the topology of software testing research over four decades of research publications. In this map, nodes represent keywords, while edges indicate that publications have co-targeted keywords. Nodes are clustered based on density and strength of edges. We examined the most common keywords, summarized clusters into research topics, examined how clusters connect, and identified emerging and declining keywords, topics, and connections.

We found that the most popular keywords tend to relate to automation, test creation and assessment guidance, assessment of system quality, and cyber–physical systems. The clusters of keywords suggest that software testing research can be divided into 16 core topics. All topics are connected, but creation guidance, automated test generation, evolution and maintenance, and test oracles have particularly strong connections to other topics, highlighting their multidisciplinary nature. Emerging keywords and topics relate to web and mobile applications, machine learning, energy consumption, automated program repair and test generation, while emerging connections have formed between web applications, test oracles, and machine learning with many topics. Random and requirements-based testing show evidence of decline.

These insights and the underlying map can inspire researchers in software testing by clarifying concepts and their relationships, or by facilitating analyses of the field (e.g., through identification of under-explored and missing connections). In future work, we will broaden the type and scope of analyses of this map data, and we make our data available so that others can do so as well.

CRediT authorship contribution statement

Alireza Salahirad: Conceptualization, Software, Investigation, Data curation, Writing – original draft, Visualization. **Gregory Gay:** Conceptualization, Methodology, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Ehsan Mohammadi:** Methodology, Validation, Writing – original draft, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Our data is provided via a link to a Zenodo repository in the manuscript.

Appendix. VOSViewer technical details

VOSviewer produces maps based on a co-occurrence matrix—a two-dimensional matrix where each column and row represents an item – a keyword, in our case – and each cell indicates the number of times two keywords co-occur. This map construction

[6] A full list of publications is maintained at https://www.vosviewer.com/publications.

[7] List of covered journals and conferences: https://www.scopus.com/sources.uri.

consists of three steps. In the first step, a similarity matrix is created from the co-occurrence matrix. A map is then formed by applying the VOS mapping technique to the similarity matrix. Finally, the map is translated, rotated, and reflected.

Forming the similarity matrix: VOSviewer takes as input a similarity matrix. This similarity matrix is obtained from the co-occurrence matrix through normalization. Normalization is done by correcting the matrix for differences in the total number of occurrences or co-occurrences of keywords. VOSviewer uses the association strength as its similarity measure (van Eck and Waltman, 2014)—in this case, the number of publications where two keywords are targeted together. Using the association strength, the similarity $s_{i,j}$ between two keywords i and j is calculated as:

$$s_{i,j} = \frac{2mc_{i,j}}{w_i w_j} \tag{A.1}$$

where m represents the total weight of all edges in the network (the total number of co-occurrences of all keywords), $c_{i,j}$ denotes the weight of the edge between keywords i and j (the total number of co-occurrences of the keywords), and w_i and w_j denote the total weight of all edges of keywords i or j (the total number of occurrences of keywords i or j and the total number of co-occurrences of these keywords with all keywords that they co-occur with). Specifically:

$$w_i = \sum_j c_{i,j} \tag{A.2}$$

$$m = \frac{1}{2}\sum_i w_i \tag{A.3}$$

The similarity between keywords i and j calculated using Eq. (A.1) is proportional to the ratio between the observed number of co-occurrences of keywords i and j and the expected number of co-occurrences of keywords i and j under the assumption that occurrences of the two keywords are independent.

Map formation: The VOS mapping technique constructs a two-dimensional map in which keywords $1, \ldots, n$ (where n is the total number of keywords) are placed such that the distance between any pair of keywords i and j reflects their similarity $s_{i,j}$ as accurately as possible. Keywords with a high similarity are located close to each other, while keywords with a low similarity are located far from each other.

The goal of the VOS mapping technique is to minimize the weighted sum of the squared Euclidean distances between all pairs of keywords (van Eck and Waltman, 2014). The higher the similarity between the two keywords, the higher the weight of their squared distance in the summation. The specific function minimized by the mapping technique is:

$$V(x_1, \ldots, x_n) = \sum_{i<j} s_{i,j} \|x_i - x_j\|^2 \tag{A.4}$$

where x_i denotes the location of keyword i in a two-dimensional space, and where $\|x_i - x_j\|$ denotes the Euclidean distance between keywords i and j. To avoid trivial maps in which all keywords have the same location, minimization is subject to the constraint that the average distance between two keywords must be equal to 1. Specifically:

$$\frac{2}{n(n-1)} \sum_{i<j} \|x_i - x_j\| = 1 \tag{A.5}$$

The constrained optimization problem – minimizing Eq. (A.4), subject to Eq. (A.5) – is solved in two steps (Van Eck and Waltman, 2010). The constrained problem is first converted into an unconstrained problem. Second, the unconstrained problem is solved using a variant of the SMACOF algorithm, an optimization algorithm commonly used in multidimensional scaling to produce human-understandable network or graph layouts through minimization of a stress function over the positions of nodes in the graph (Borg and Groenen, 2005).

Clustering of Keywords: Keywords are assigned to clusters, and the number of clusters is determined, through optimization. This is a common approach for clustering nodes in a network (Newman, 2004). Potential assignments of keywords to clusters are assessed using the function:

$$V(c_1, \ldots, c_n) = \sum_{i<j} \delta(c_i, c_j)(s_{i,j} - \gamma) \tag{A.6}$$

where c_i is the cluster that keyword i has been assigned to. $\delta(c_i, c_j)$ is a difference function that yields 1 if $c_i = c_j$ and 0, otherwise. γ determines the level of clustering, with higher values yielding a larger number of clusters. This equation is unified with the mapping function minimized in Eq. (A.4), and includes the same similarity measurement $s_{i,j}$ calculated in Eq. (A.1).

There is no maximum number of keywords per cluster. The minimum number of keywords is controlled using a user-specified parameter. We used the default, a minimum of one keyword. The clustering algorithm will merge small clusters in cases where merging does not affect the result of Eq. (A.6). Therefore, any small cluster that remain are ones that affect the results of the equation.

Eq. (A.6) is maximized using the smart local moving algorithm (Waltman and van Eck, 2013). Following the optimization, the assignment of keywords to clusters that maximizes Eq. (A.6) is returned. This process yields a small number of clusters containing keywords that are targeted disproportionately often together in publications.

Translation, rotation, and reflection: The optimization problem introduced in Eq. (A.4) does not have a single global optimal solution (Van Eck and Waltman, 2010). However, consistent results are desirable. To ensure that the same co-occurrence matrix always yields the same map, three transformations are applied after optimization:

- **Translation:** The solution is translated to be centered at the origin.
- **Rotation:** Principle component analysis is applied in order to maximize variance on the horizontal dimension.
- **Reflection:** If the median of $x_{1,1}, \ldots, x_{n,1}$ is larger than 0, the solution is reflected in the vertical axis. If the median of $x_{1,2}, \ldots, x_{n,2}$ is larger than 0, the solution is reflected in the horizontal axis.

References

Ahmed, B.S., Enoiu, E., Afzal, W., Zamli, K.Z., 2020. An evaluation of Monte Carlo-based hyper-heuristic for interaction testing of industrial embedded software applications. Soft Comput. 24 (18), 13929–13954.

Alshahwan, N., Gao, X., Harman, M., Jia, Y., Mao, K., Mols, A., Tei, T., Zorin, I., 2018. Deploying search based software engineering with sapienz at facebook. In: Search-Based Software Engineering. Springer International Publishing, Cham, pp. 3–45.

Anand, S., Burke, E.K., Chen, T.Y., Clark, J., Cohen, M.B., Grieskamp, W., Harman, M., Harrold, M.J., McMinn, P., 2013. An orchestrated survey of methodologies for automated software test case generation. J. Syst. Softw. 86 (8), 1978–2001.

Arcuri, A., Briand, L.C., 2011. Adaptive random testing: An illusion of effectiveness? In: ISSTA.

Barr, E., Harman, M., McMinn, P., Shahbaz, M., Yoo, S., 2015. The oracle problem in software testing: A survey. IEEE Trans. Softw. Eng. 41 (5), 507–525. http://dx.doi.org/10.1109/TSE.2014.2372785.

Bass, L., Clements, P., Kazman, R., 1998. Software Architecture in Practice. Addison-Wesley.

Bertolino, A., 2007. Software testing research: Achievements, challenges, dreams. In: Future of Software Engineering (FOSE '07). pp. 85–103. http://dx.doi.org/10.1109/FOSE.2007.25.

Borg, I., Groenen, P.J., 2005. Modern Multidimensional Scaling: Theory and Applications. Springer Science & Business Media.

Borgman, C.L., Furner, J., 2002. Scholarly communication and bibliometrics. Annu. Rev. Inf. Sci. Technol. 36 (1), 2–72. http://dx.doi.org/10.1002/aris.1440360102, URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440360102. arXiv:https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/aris.1440360102.

Bozkurt, M., Harman, M., Hassoun, Y., et al., 2010. Testing Web Services: A Survey. Tech. Rep. TR-10-01, Department of Computer Science, King's College London.

Cadar, C., Dunbar, D., Engler, D., 2008. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation. OSDI '08, USENIX Association, Berkeley, CA, USA, pp. 209–224, URL: http://dl.acm.org/citation.cfm?id=1855741.1855756.

Catal, C., 2011. Software fault prediction: A literature review and current trends. Expert Syst. Appl. 38 (4), 4626–4636.

Cavacini, A., 2015. What is the best database for computer science journal articles? Scientometrics 102 (3), 2059–2071. http://dx.doi.org/10.1007/s11192-014-1506-1.

Chen, T.Y., Kuo, F.-C., Merkel, R.G., Tse, T., 2010. Adaptive random testing: The art of test case diversity. J. Syst. Softw. 83 (1), 60–66.

Chen, J., Patra, J., Pradel, M., Xiong, Y., Zhang, H., Hao, D., Zhang, L., 2020. A survey of compiler testing. ACM Comput. Surv. 53 (1), 1–36.

Clarke, E.M., Emerson, E.A., Sistla, A.P., 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 244–263.

Copeland, L., 2004. A Practitioner's Guide to Software Test Design. Artech House.

Crossley, M., 2000. The Desk Reference of Statistical Quality Methods. ASQ Quality Press.

De Bellis, N., 2009. Bibliometrics and Citation Analysis: From the Science Citation Index to Cybermetrics. scarecrow Press.

de Freitas, F.G., de Souza, J.T., 2011. Ten years of search based software engineering: A bibliometric analysis. In: Cohen, M.B., Ó Cinnéide, M. (Eds.), Search Based Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 18–32.

Ding, Y., Liu, X., Guo, C., Cronin, B., 2013. The distribution of references across texts: Some implications for citation analysis. J. Informetr. 7 (3), 583–592. http://dx.doi.org/10.1016/j.joi.2013.03.003, URL: https://www.sciencedirect.com/science/article/pii/S1751157713000230.

Donthu, N., Kumar, S., Pattnaik, D., 2020. Forty-five years of journal of business research: a bibliometric analysis. J. Bus. Res. 109, 1–14.

Douglas-Smith, D., Iwanaga, T., Croke, B.F., Jakeman, A.J., 2020. Certain trends in uncertainty and sensitivity analysis: An overview of software tools and techniques. Environ. Model. Softw. 124, 104588.

Farhoodi, R., Garousi, V., Pfahl, D., Sillito, J., 2013. Development of scientific software: A systematic mapping, a bibliometrics study, and a paper repository. Int. J. Softw. Eng. Knowl. Eng. 23 (04), 463–506. http://dx.doi.org/10.1142/S0218194013500137, arXiv:https://doi.org/10.1142/S0218194013500137.

Fewster, M., Graham, D., 1999. Software Test Automation. Addison-Wesley Reading.

Fontes, A., Gay, G., 2021. Using machine learning to generate test oracles: A systematic literature review. In: Proceedings of the 1st International Workshop on Test Oracles. In: TORACLE 2021, Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/3472675.3473974.

Fortunato, S., Bergstrom, C.T., Börner, K., Evans, J.A., Helbing, D., Milojević, S., Petersen, A.M., Radicchi, F., Sinatra, R., Uzzi, B., Vespignani, A., Waltman, L., Wang, D., Barabási, A.-L., 2018. Science of science. Science 359 (6379), http://dx.doi.org/10.1126/science.aao0185, URL: https://science.sciencemag.org/content/359/6379/eaao0185. arXiv:https://science.sciencemag.org/content/359/6379/eaao0185.full.pdf.

France, R., Rumpe, B., 2007. Model-driven development of complex systems: A research roadmap. In: Briand, L., Wolf, A. (Eds.), Future of Software Engineering 2007. IEEE-CS Press.

Fukuda, T., 1992. Theory and applications of neural networks for industrial control systems. IEEE Trans. Ind. Electron. 472–489.

Galeotti, J.P., Fraser, G., Arcuri, A., 2013. Improving search-based test suite generation with dynamic symbolic execution. In: 2013 IEEE 24th International Symposium on Software Reliability Engineering. ISSRE, pp. 360–369. http://dx.doi.org/10.1109/ISSRE.2013.6698889.

Garousi, V., 2015. A bibliometric analysis of the Turkish software engineering research community. Scientometrics 105 (1), 23–49. http://dx.doi.org/10.1007/s11192-015-1663-x.

Garousi, V., Fernandes, J.M., 2017. Quantity versus impact of software engineering papers: a quantitative study. Scientometrics 112 (2), 963–1006. http://dx.doi.org/10.1007/s11192-017-2419-6.

Garousi, V., Mäntylä, M.V., 2016. Citations, research topics and active countries in software engineering: A bibliometrics study. Comp. Sci. Rev. 19, 56–77. http://dx.doi.org/10.1016/j.cosrev.2015.12.002, URL: https://www.sciencedirect.com/science/article/pii/S1574013715300654.

Garousi, V., Varma, T., 2010. A bibliometric assessment of canadian software engineering scholars and institutions (1996–2006). Comput. Inf. Sci. 3 (2), 19.

Gay, G., Rayadurgam, S., Heimdahl, M.P.E., 2017. Automated steering of model-based test oracles to admit real program behaviors. IEEE Trans. Softw. Eng. 43 (6), 531–555. http://dx.doi.org/10.1109/TSE.2016.2615311.

Gay, G., Staats, M., Whalen, M., Heimdahl, M., 2015. The risks of coverage-directed test case generation. Softw. Eng. IEEE Trans. PP (99), http://dx.doi.org/10.1109/TSE.2015.2421011.

Harrold, M.J., 2000. Testing: A roadmap. In: Proceedings of the Conference on the Future of Software Engineering. ICSE '00, Association for Computing Machinery, New York, NY, USA, pp. 61–72. http://dx.doi.org/10.1145/336512.336532.

Helali Moghadam, M., Saadatmand, M., Borg, M., Bohlin, M., Lisper, B., 2019. Machine learning to guide performance testing: An autonomous test framework. In: 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops. ICSTW, pp. 164–167.

Herzner, W., Schlick, R., Le Guennec, A., Martin, B., 2007. Model-based simulation of distributed real-time applications. In: 5th IEEE Int'l Conf. on Industrial Infomatics. pp. 989–994.

Hong, S., Ahn, J., Park, S., Kim, M., Harrold, M.J., 2012. Testing concurrent programs to achieve high synchronization coverage. In: Proceedings of the 2012 International Symposium on Software Testing and Analysis. pp. 210–220.

Jackson, D., 2019. Alloy: a language and tool for exploring software designs. Commun. ACM 62 (9), 66–76.

Jamali, H., Steel, C., Mohammadi, E., 2020. Wine research and its relationship with wine production: a scientometric analysis of global trends. Aust. J. Grape Wine Res. 26 (2), 130–138. http://dx.doi.org/10.1111/ajgw.12422, URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/ajgw.12422. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/ajgw.12422.

Jamil, M.A., Arif, M., Abubakar, N.S.A., Ahmad, A., 2016. Software testing techniques: A literature review. In: 2016 6th International Conference on Information and Communication Technology for the Muslim World (ICT4M). IEEE, pp. 177–182.

Just, R., 2014. The major mutation framework: Efficient and scalable mutation analysis for Java. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis. In: ISSTA 2014, ACM, New York, NY, USA, pp. 433–436. http://dx.doi.org/10.1145/2610384.2628053, URL: http://doi.acm.org/10.1145/2610384.2628053.

Just, R., Schweiggert, F., Kapfhammer, G.M., 2011. MAJOR: An efficient and extensible tool for mutation analysis in a java compiler. In: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. ASE '11, IEEE Computer Society, Washington, DC, USA, pp. 612–615. http://dx.doi.org/10.1109/ASE.2011.6100138.

Karanatsiou, D., Li, Y., Arvanitou, E.-M., Misirlis, N., Wong, W.E., 2019. A bibliometric assessment of software engineering scholars and institutions (2010–2017). J. Syst. Softw. 147, 246–261. http://dx.doi.org/10.1016/j.jss.2018.10.029, URL: https://www.sciencedirect.com/science/article/pii/S0164121218302334.

Kitchenham, B., Pfleeger, S., 1996. Software quality: the elusive target [special issues section]. IEEE Softw. 13 (1), 12–21. http://dx.doi.org/10.1109/52.476281.

Korel, B., Al-Yami, A.M., 1998. Automated regression test generation. In: Proceedings of the 1998 ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA '98, ACM, New York, NY, USA, pp. 143–152. http://dx.doi.org/10.1145/271771.271803, URL: http://doi.acm.org/10.1145/271771.271803.

Li, H., An, H., Wang, Y., Huang, J., Gao, X., 2016. Evolutionary features of academic articles co-keyword network and keywords co-occurrence network: Based on two-mode affiliation network. Physica A 450, 657–669. http://dx.doi.org/10.1016/j.physa.2016.01.017, URL: https://www.sciencedirect.com/science/article/pii/S037843711600025X.

Liao, H., Tang, M., Luo, L., Li, C., Chiclana, F., Zeng, X.-J., 2018. A bibliometric analysis and visualization of medical big data research. Sustainability 10 (1), http://dx.doi.org/10.3390/su10010166, URL: https://www.mdpi.com/2071-1050/10/1/166.

Liu, Y., Goncalves, J., Ferreira, D., Xiao, B., Hosio, S., Kostakos, V., 2014. CHI 1994–2013: Mapping two decades of intellectual progress through co-word analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '14, Association for Computing Machinery, New York, NY, USA, pp. 3553–3562. http://dx.doi.org/10.1145/2556288.2556969.

Martinez, M., Durieux, T., Sommerard, R., Xuan, J., Monperrus, M., 2017. Automatic repair of real bugs in java: a large-scale experiment on the defects4j dataset. Empir. Softw. Eng. 22 (4), 1936–1964. http://dx.doi.org/10.1007/s10664-016-9470-4.

Marx, W., Haunschild, R., Bornmann, L., 2017. Global warming and tea production—The bibliometric view on a newly emerging research topic. Climate 5 (3), http://dx.doi.org/10.3390/cli5030046, URL: https://www.mdpi.com/2225-1154/5/3/46.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, USA.
Moed, H.F., 2006. Citation Analysis in Research Evaluation, Vol. 9. Springer Science & Business Media.
Mohammadi, E., 2012. Knowledge mapping of the Iranian nanoscience and technology: a text mining approach. Scientometr. Scientometr. 92 (3), 593–608. http://dx.doi.org/10.1007/s11192-012-0644-6, URL: https://akjournals.com/view/journals/11192/92/3/article-p593.xml.
Mohammadi, E., Karami, A., 2020. Exploring research trends in big data across disciplines: A text mining analysis. J. Inf. Sci. 0165551520932855. http://dx.doi.org/10.1177/0165551520932855, arXiv:https://doi.org/10.1177/0165551520932855.
Mok, A.K., Stuart, D., 1996. Simulation vs. verification: Getting the best of both worlds. In: Proceedings of the Eleventh Annual Conf. on Computer Assurance, COMPASS 96.
Morell, L., 1990. A theory of fault-based testing. IEEE Trans. Softw. Eng. 16 (8), 844–857.
Murphy-Hill, E., Parnin, C., Black, A.P., 2012. How we refactor, and how we know it. IEEE Trans. Softw. Eng. 38 (1), 5–18. http://dx.doi.org/10.1109/TSE.2011.41.
Newman, M.E., 2004. Fast algorithm for detecting community structure in networks. Phys. Rev. E 69 (6), 066133.
Nie, C., Leung, H., 2011. A survey of combinatorial testing. ACM Comput. Surv. 43 (2), http://dx.doi.org/10.1145/1883612.1883618.
Orso, A., Rothermel, G., 2014. Software testing: A research travelogue (2000–2014). In: Proceedings of the on Future of Software Engineering. In: FOSE 2014, ACM, New York, NY, USA, pp. 117–132. http://dx.doi.org/10.1145/2593882.2593885, URL: http://doi.acm.org/10.1145/2593882.2593885.
Peters, H., van Raan, A., 1993. Co-word-based science maps of chemical engineering. Part I: Representations by direct multidimensional scaling. Res. Policy 22 (1), 23–45. http://dx.doi.org/10.1016/0048-7333(93)90031-C, URL: https://www.sciencedirect.com/science/article/pii/004873339390031C.
Pezze, M., Young, M., 2006. Software Test and Analysis: Process, Principles, and Techniques. John Wiley and Sons.
Pritchard, A., et al., 1969. Statistical bibliography or bibliometrics. J. Doc. 25 (4), 348–349.
Qureshi, I.A., Nadeem, A., 2013. GUI testing techniques: a survey. Int. J. Future Comput. Commun. 2 (2), 142.
Romo-Fernández, L.M., Guerrero-Bote, V.P., Moya-Anegón, F., 2013. Co-word based thematic analysis of renewable energy (1990–2010). Scientometrics 97 (3), 743–765.
Rothermel, G., Harrold, M.J., 1996. Analyzing regression test selection techniques. IEEE Trans. Softw. Eng. 22 (8), 529–551.
Rothermel, G., Harrold, M.J., von Ronne, J., Hong, C., 2002. Empirical studies of test-suite reduction. Softw. Test. Verif. Reliab. 12 (4), 219–249. http://dx.doi.org/10.1002/stvr.256.
RTCA/DO-178C, 2012. Software considerations in airborne systems and equipment certification.
Rumbaugh, J., Jacobson, I., Booch, G., 1998. The Unified Modeling Language Reference Manual. In: Object Technology Series, Addison-Wesley.
Salahirad, A., Almulla, H., Gay, G., 2019. Choosing the fitness function for the job: Automated generation of test suites that detect real faults. Softw. Test. Verif. Reliab. 29 (4–5), e1701. http://dx.doi.org/10.1002/stvr.1701, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1701. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.1701, e1701 stvr.1701.
Su, H.-N., Lee, P.-C., 2010. Mapping knowledge structure by keyword co-occurrence: a first look at journal papers in Technology Foresight. Scientometrics 85 (1), 65–79.
Su, T., Wu, K., Miao, W., Pu, G., He, J., Chen, Y., Su, Z., 2017. A survey on data-flow testing. ACM Comput. Surv. 50 (1), 1–35.
Takanen, A., DeMott, J., Miller, C., 2008. Fuzzing for Software Security Testing and Quality Assurance. Artech House, Inc..
Thelwall, M., Sud, P., 2022. Scopus 1900–2020: Growth in articles, abstracts, countries, fields, and journals. Quantitative Science Studies 1–14. http://dx.doi.org/10.1162/qss_a_00177, arXiv:https://direct.mit.edu/qss/article-pdf/doi/10.1162/qss_a_00177/1992507/qss_a_00177.pdf.
Turhan, B., Menzies, T., Bener, A.B., Di Stefano, J., 2009. On the relative value of cross-company and within-company data for defect prediction. Empir. Softw. Eng. 14 (5), 540–578.
Turing, A., 1989. Checking a large routine. In: The Early British Computer Conferences. MIT Press, Cambridge, MA, USA, pp. 70–72.
Van Eck, N.J., Waltman, L., 2010. Software survey: VOSviewer, a computer program for bibliometric mapping. Scientometrics 84 (2), 523–538.
Van Eck, N.J., Waltman, L., 2014. Visualizing bibliometric networks. In: Measuring Scholarly Impact. Springer, pp. 285–320.
van Eck, N.J., Waltman, L., 2014. Visualizing bibliometric networks. In: Measuring Scholarly Impact: Methods and Practice. Springer International Publishing, Cham, pp. 285–320. http://dx.doi.org/10.1007/978-3-319-10377-8_13.
van Lamsweerde, A., 2007. Engineering requirements for system reliability and security. Softw. Syst. Reliab. Secur. 9.
Voas, J., 1997. Fault injection for the masses. Computer 30 (12), 129–130. http://dx.doi.org/10.1109/2.642820.
Waltman, L., van Eck, N.J., 2013. A smart local moving algorithm for large-scale modularity-based community detection. Eur. Phys. J. B 86 (11), 471. http://dx.doi.org/10.1140/epjb/e2013-40829-0.
Weyuker, E., Jeng, B., 1991. Analyzing partition testing strategies. IEEE Trans. Softw. Eng. 17 (7), 703–711.
Whalen, M., Rajan, A., Heimdahl, M., 2006. Coverage metrics for requirements-based testing. In: Proceedings of Int'l Symposium on Software Testing and Analysis. ACM, pp. 25–36.
Whittaker, E., 1989. A History of the Theories of Aether and Electricity: Vol. I: The Classical Theories; Vol. II: The Modern Theories, 1900-1926, Vol. 1. Courier Dover Publications.
Wong, W.E., Tse, T., Glass, R.L., Basili, V.R., Chen, T., 2008. An assessment of systems and software engineering scholars and institutions (2001–2005). J. Syst. Softw. 81 (6), 1059–1062. http://dx.doi.org/10.1016/j.jss.2007.09.018, URL: https://www.sciencedirect.com/science/article/pii/S0164121207002300, Agile Product Line Engineering.
Wong, W.E., Tse, T., Glass, R.L., Basili, V.R., Chen, T., 2009. An assessment of systems and software engineering scholars and institutions (2002–2006). J. Syst. Softw. 82 (8), 1370–1373. http://dx.doi.org/10.1016/j.jss.2009.06.018, URL: https://www.sciencedirect.com/science/article/pii/S0164121209001265, SI: Architectural Decisions and Rationale.
Wong, W.E., Tse, T., Glass, R.L., Basili, V.R., Chen, T., 2011. An assessment of systems and software engineering scholars and institutions (2003–2007 and 2004–2008). J. Syst. Softw. 84 (1), 162–168. http://dx.doi.org/10.1016/j.jss.2010.09.036, URL: https://www.sciencedirect.com/science/article/pii/S0164121210002682, Information Networking and Software Services.
Yoo, S., Harman, M., 2012. Regression testing minimization, selection and prioritization: a survey. Softw. Test. Verif. Reliab. 22 (2), 67–120.
Zander, J., Schieferdecker, I., Mosterman, P.J., 2011. Model-Based Testing for Embedded Systems. CRC Press.
Zeller, A., 2001. Automated debugging: Are we close? Computer 34 (11), 26–31.
Zeller, A., Hildebrandt, R., 2002. Simplifying and isolating failure-inducing input. IEEE Trans. Softw. Eng. 28 (2), 183–200. http://dx.doi.org/10.1109/32.988498.

Alireza Salahirad is a Ph.D. candidate in the Department of Computer Science and Engineering at the University of South Carolina, under the supervision of Dr. Gregory Gay. He is interested in software testing, mutation testing, automated test generation, and source code metrics.

Gregory Gay is an Associate Professor in the Department of Computer Science and Engineering at Chalmers and the University of Gothenburg. His research interests include software testing and analysis and search-based software engineering. Dr. Gay's research blends optimization, artificial intelligence, and machine learning to help developers deliver complex systems in a safe, secure, and efficient manner.

Ehsan Mohammadi is an Assistant Professor at the University of South Carolina School of Information Science. He is interested in applying computational methods to answer social sciences research questions in different areas, including Science of Science and Social Media Analytics, among others.