Artificial Intelligence Applied to Software Testing: A Tertiary Study

DOMENICO AMALFITANO, University of Naples Federico II, Italy
STEFANO FARALLI, Sapienza University of Rome, Italy
JEAN CARLO ROSSA HAUCK, Federal University of Santa Catarina, Brazil
SANTIAGO MATALONGA, University of the West of Scotland, United Kingdom
DAMIANO DISTANTE, University of Rome UnitelmaSapienza, Italy

Context: Artificial intelligence (AI) methods and models have extensively been applied to support different phases of the software development lifecycle, including software testing (ST). Several secondary studies investigated the interplay between AI and ST but restricted the scope of the research to specific domains or sub-domains within either area.
Objective: This research aims to explore the overall contribution of AI to ST, while identifying the most popular applications and potential paths for future research directions.
Method: We executed a tertiary study following well-established guidelines for conducting systematic literature mappings in software engineering and for answering nine research questions.
Results: We identified and analyzed 20 relevant secondary studies. The analysis was performed by drawing from well-recognized AI and ST taxonomies and mapping the selected studies according to them. The resulting mapping and discussions provide extensive and detailed information on the interplay between AI and ST.
Conclusion: The application of AI to support ST is a well-consolidated and growing interest research topic. The mapping resulting from our study can be used by researchers to identify opportunities for future research, and by practitioners looking for evidence-based information on which AI-supported technology to possibly adopt in their testing processes.

CCS Concepts: • **Software and its engineering → Software creation and management**; **Software verification and validation**; **Software testing and debugging**; • **Computing methodologies → Artificial intelligence**; **Machine learning**; • **General and reference → Surveys and overviews;**

Additional Key Words and Phrases: Artificial intelligence, Software testing, Taxonomy, Tertiary study, Systematic literature review, Systematic mapping study

ACM Reference format:
Domenico Amalfitano, Stefano Faralli, Jean Carlo Rossa Hauck, Santiago Matalonga, and Damiano Distante. 2023. Artificial Intelligence Applied to Software Testing: A Tertiary Study. *ACM Comput. Surv.* 56, 3, Article 58 (October 2023), 38 pages.
https://doi.org/10.1145/3616372

Authors' addresses: D. Amalfitano, University of Naples Federico II, Via Claudio, 21, Napoli, Italy, 80125; email: domenico.amalfitano@unina.it; S. Faralli, Sapienza University of Rome, Via Salaria, 113, Rome, Italy, 00198; email: faralli@di.uniroma1.it; J. C. R. Hauck, Federal University of Santa Catarina, Campus Universitário Trindade, Florianopolis, Brazil, 88040-200; email: jean.hauck@ufsc.br; S. Matalonga, University of the West of Scotland, High Street Paisley, Paisley, United Kingdom, PA13AP; email: santiago.matalonga@uws.ac.uk; D. Distante, University of Rome UnitelmaSapienza, Piazza Sassari, 4, Rome, Italy, 00161; email: damiano.distante@unitelmasapienza.it.

0360-0300/2023/10-ART58 $15.00
https://doi.org/10.1145/3616372

1 INTRODUCTION

Software testing **(ST)** and ***artificial intelligence*** **(AI)** are two research areas with a long and ripe history in computing. AI methodologies and techniques have been around for more than 50 years [38] and, in the current century, with the advances in computational resources and the abundance of data, their potential has vastly increased. As a consequence, AI has been applied to fields as diverse as healthcare [39], project management [54], finance [66], law [93], and many more. Both the academic research community and the industry have injected AI paradigms to provide solutions to traditional engineering problems. Similarly, AI has evidently been useful to **software engineering (SE)** [7, 13, 47]. ST has always been an intrinsic part of the software development lifecycle [85]. Yet, as software has become more and more pervasive, it has also grown in size and complexity [16], bringing new challenges to software testing practices [64]. Therefore, with AI poised to enhance knowledge work, there is interest in analyzing how it has been used to improve testing practices. Several studies have explored the interplay between AI and ST [51]. Yet, given the breadth and depth of each of these disciplines, high-quality review studies tend to focus their scope on orthogonal selections in each of these areas. For instance, the use of evolutionary algorithms for regression test case prioritization has been investigated in [76] while the application of natural language processing technique in ST has been analyzed in [35]. Alternatively, unstructured review papers or position papers have proposed how these two fields would merge.

The goal of this work is to uncover evidence on how AI has been applied to support ST, to reveal established hinge points between the two research areas and future trends. In particular, in this study, we focus on dynamic testing that, according to the ISO 29119 standard [34], comprises the activities that are performed to assess whether a software product works as expected when it is executed. To achieve this goal, we conducted a tertiary systematic mapping study. In this work, we adhere to the definitions by Kitchenham et al. [52]. A primary study is an empirical investigation into specific research questions, while a secondary study is a review of primary studies related to specific research questions with the aim of synthesising evidence. Finally, a tertiary study, of which this research is an example, is a review of secondary studies related to the same research questions with the aim of uncovering mappings and/or trends. The need for a tertiary study, particularly a systematic mapping study, on the interplay of AI and ST is motivated by the following considerations:

- although there are already several secondary studies investigating the application of AI to ST, to manage the vastness of the two research areas, most of these studies limit their scope with an orthogonal division of one or both areas;
- there is a wealth of primary studies that makes it unfeasible to approach our research goal with a secondary study, if not by limiting the scope of the research, as the identified secondary studies have done;
- a systematic process makes the work reproducible and provides internal consistency to the results and focuses the discussion on available evidence in existing secondary studies;
- a systematic mapping study is suited to structure a research area [77], and as such is more suitable than a systematic literature review in our research context because of the size and scope of the bodies of knowledge (AI and ST). Furthermore, after an initial investigation, we noted that secondary studies that applied systematic literature reviews as their research method are able to do so by limiting the scope to a sub-domain (for instance, search base techniques for ST). Therefore, to observe the whole possible interplay between AI and ST, a systematic mapping is the suitable research method for this tertiary study.

As a main contribution, this article provides a broad view of the AI applications in support of ST. An additional novel contribution of this work is a fine-grained mapping showing how specific

testing fields have been supported by specific AI sub-domains and methodologies. This mapping can be used by researchers to identify open topics for future research on new applications of AI for ST and by practitioners to make decisions on the most suitable AI-supported technologies that can be introduced in their testing processes. To the best of our knowledge, this is the first tertiary study that attempts to depict a comprehensive picture of how AI is used to support ST, and how the two research domains are woven. The remainder of the article is organized as follows. Section 2 introduces the key concepts and terminology related to the areas of interest of our study. Section 3 describes the protocol we designed to support the process of selecting secondary studies of interest and for extracting evidence from them. Section 4 provides insights about the process execution. Section 5 analyzes extracted data and answers our research questions. Section 6 presents overall considerations on the results of our study and provides a focus on testing activities whose automation has been supported by different AI techniques. Section 7 discusses threats to the validity of our study. Finally, Section 8 concludes the article and provides final remarks.

2 BACKGROUND

AI and ST are two large and complex research areas for which there are no universally agreed upon taxonomies nor bodies of knowledge. As a way to define the language and vocabulary that has been used throughout the article, we built two taxonomies, one for each research area. The taxonomy shown in Figure 6 reports the AI key concepts that have been used to support ST, whereas the one in Figure 7 refers to the ST key concepts that have been supported by AI. In the following sections, we provide a short description of the two research areas, the domains, and sub-domains of each taxonomy along with a short definition of related key concepts that are relevant to our study. For each key concept, we also provide a proper literature reference from which it is possible to access more detailed and complete definitions.

2.1 Artificial Intelligence

Although there exist many definitions of AI, for the aims of this study, we mention the one given in the European Commission JCR report on AI [59]: *"AI is a generic term that refers to any machine or algorithm that is capable of observing its environment, learning, and based on the knowledge and experience gained, taking intelligent action or proposing decisions. There are many technologies that fall under this broad AI definition. At the moment, ML techniques are the most widely used."* This definition was adopted by the *AI Watch*[1] in [88] as the starting point for the specification of an operational definition and a taxonomy of AI aimed at supporting the mapping of the AI landscape and at detecting AI applications in a wide range of technological contexts. The taxonomy provided by the AI Watch report includes five core scientific domains, namely, *Reasoning*, *Planning*, *Learning*, *Communication*, and *Perception*, and three transversal domains, namely *Integration and Interaction*, *Services*, and *Ethics and Philosophy*. The overall taxonomy is depicted in Figure 6, where: (i) white boxes represent domains and key concepts drawn from the AI Watch report [88], while (ii) gray boxes are additional key concepts extracted, during the mapping process, from the analyzed secondary studies.

2.1.1 Reasoning. The AI domain studying methodologies to transform data into knowledge and infer facts from them. This domain includes three sub-domains: *knowledge representation*, *automated reasoning*, and *common sense reasoning*. *Knowledge representation* is the area of AI addressing the problem of representing, maintaining, and manipulating knowledge [56]. *Automated*

[1]The European Commission knowledge service to monitor the development, uptake, and impact of artificial intelligence for Europe.

reasoning is concerned with the study of using algorithms that allow machines to reason automatically [14]. Finally, as described in [27], *common sense reasoning* is the field of science studying the human-like ability to make presumptions about the type and essence of ordinary situations. Key concepts related to our study and belonging to this domain are: (i) *fuzzy logic*, a form of logic in which the truth value of variables may be any real number (between 0 and 1) [72], (ii) *knowledge representation and reasoning*, the use of symbolic rules to represent and infer knowledge [56], (iii) *ontologies*, forms of knowledge representation facilitating knowledge sharing and reuse [31], and (iv) *semantic web*, an extension of the World Wide Web through standards set by the World Wide Web Consortium,[2] which *"…enables people to create data stores on the Web, build vocabularies, and write rules for handling data"*[3] [75].

2.1.2 Planning. The AI domain whose main purpose concerns the design and execution of strategies to carry out an activity, typically performed by intelligent agents, autonomous robots, and unmanned vehicles. In this domain, strategies are identified by complex solutions that must be discovered and optimized in a multidimensional space. This domain includes three highly related sub-domains dealing with the problem of optimizing the search for solutions to planning and scheduling problems, namely, *planning and scheduling*, *searching*, and *optimization*. Key concepts related to our study and belonging to this domain are: (i) *constraint satisfaction*, the process of finding a solution to a set of constraints on a set of variables [99], (ii) *evolutionary algorithms*, a subset of metaheuristic optimization algorithms based on mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection [6], (iii) *genetic algorithms*, a branch of evolutionary algorithms inspired by the process of natural selection relying on biologically inspired operators such as mutation, crossover and selection [69], (iv) *graph plan algorithms*, a family of planning algorithms based on the expansion of compact structures known as planning graphs [17], (v) *hyper-heuristics*, the field dealing with the problem of automating the design of heuristic methods to solve hard computational search problems [21], and (vi) *metaheuristic optimization*, the research field dealing with optimization problems using metaheuristic algorithms [19].

2.1.3 Learning. The AI domain dealing with the ability of systems to automatically learn, decide, predict, adapt and react to changes and improve from experience, without being explicitly programmed. The corresponding branch of the resulting taxonomy is mainly constructed with *machine learning* (ML)-related concepts. Key concepts related to our study and belonging to this domain are: (i) *artificial neural networks*, a family of supervised algorithms inspired by the biological neural networks that constitute animal brains [41], the training of a neural network consists in observing the data regarding the inputs and the expected output, and in forming probability-weighted associations between the two, which are stored within the data structure of the network itself - designed as a sequence of layers of connected perceptrons [86], (ii) *boosting*, is an ensemble meta-algorithm for the reduction of bias and variance error's components [20], (iii) *classification*, a supervised task where a model is trained on a population of instances labeled with a discrete set of labels and the outcome is a set of predicted labels for a given collection of unobserved instances [55], (iv) *clustering*, an unsupervised task were given a similarity function, objects are grouped into clusters so that objects in the same cluster are more similar to each other than to objects in other clusters [105], (v) *convolutional neural networks*, a specialized type of neural networks that uses convolution in place of general matrix multiplication in at least one of its layers [37], (vi) *decision trees*, a family of classification and regression algorithms that learn hierarchical structures of simple decision rules from data and whose resulting models can be depicted as trees were nodes represent

[2]W3C https://www.w3.org/
[3]SEMANTIC WEB-W3C https://www.w3.org/standards/semanticweb/

decision rules and leaf nodes are the outcomes [70], (vii) *ensemble methods*, algorithms leveraging a set of individually trained classifiers (such as, decision trees) whose predictions are combined to produce more accurate predictions than any of the single classifiers [73], (viii) *probabilistic models*, a family of classifiers that are able to predict, given an observation of an input, a probability distribution over a set of classes [40], (ix) *recurrent neural networks*, neural networks with recurrent connections, which can be used to map input sequences to output sequences [15], (x) *reinforcement learning*, is one of the fundamental machine learning paradigms, where algorithms address the "problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment" [46], (xi) *regression*, a set of mathematical methods that allow data scientists to predict a continuous outcome based on the value of one or more predictor variables [106], (xii) *supervised learning*, a machine learning paradigm for problems where the available data consists of labelled examples [87], (xiii) *support vector machines*, supervised learning algorithms where input features are non-linearly mapped to a very high-dimension feature space and a linear decision surface is constructed to generate classification and regression analysis models [24], and (xiv) *unsupervised learning*, one of the fundamental machine learning paradigms where algorithms try to learn patterns from unlabelled data [87].

2.1.4 Communication. The AI domain referring to the abilities of identifying, processing, understanding, and generating information from written and spoken human communications. This domain is mainly covered by the *natural language processing* (NLP) [45, 62]. Key concepts related to our study and belonging to this domain are: (i) *information extraction*, the automatic extraction of structured information, such as entities, relationships and attributes describing entities, from unstructured sources [90], (ii) *information retrieval* deals with the problem of *"finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)"* [61], (iii) *natural language generation* refers to *"the process of constructing natural language outputs from non-linguistic inputs"* [80], (iv) *natural language understanding* refers to *"computer understanding of human language, which includes spoken as well as typed communication"* [103], (v) *text mining* is the semi-automated process of extracting knowledge from a large number of unstructured texts [29], and (vi) *word embedding* is *"a word representation involving the mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension"* [45].

2.1.5 Perception. Refers to the ability of a system to become aware of the environment through the senses of vision and hearing. Although this is a broad domain of AI with many AI applications, the only key concept coming from this domain (particularly, from the sub-domain of *computer vision*) related to our study is *image processing*, which is the field dealing with the use of machines to process digital images through algorithms [78].

2.1.6 Integration and Interaction. A transversal AI domain comprising, among others, the *multi-agent systems* sub-domain. It can be described as the domain that addresses the combination of perception, reasoning, action, learning and interaction with the environment, as well as characteristics such as distribution, coordination, cooperation, autonomy, interaction and integration. Key concepts related to our study and belonging to this domain are: (i) *intelligent agent*, an entity equipped with sensors and actuators that exhibits some form of intelligence in its action and thought [87], (ii) *q-learning*, a *reinforcement learning* algorithm (see Section 2.1.3), which provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains [102], and (iii) *swarm intelligence*, which refers to the algorithms typically based on a population of simple agents interacting locally with one another and with their environment [42].

For completeness, we remark that the reference AI Watch taxonomy includes two—unrelated to this study—additional transversal domains, namely, *Services* and *Ethics and Philosophy*. Where, the first domain includes all forms of infrastructure, software and platform provided as services or applications, and the second is related to important issues regarding the impact of AI technologies in our society.

2.2 Software Testing

ST is defined by the *29119-1-2013 ISO/IEC/IEEE International Standard* as *a process made by a set of interrelated or interacting activities aimed at providing two types of confirmations: verification and validation [34]*. Verification is a confirmation that specified requirements have been fulfilled in a given software product (a.k.a., work item or test item), whereas validation demonstrates that the work item can be adopted by the users for their specific tasks. The main objective of ST is to assess the absence of *faults*, *errors*, or *failures* in the test items. Among the great number of taxonomies proposed in the literature for describing the different and heterogeneous aspects of the ST research area, in this work, we refer to the unified view proposed by Software Engineering Body of Knowledge (SWEBOK) [18]. The SWEBOK is a guide to the broad scope of software engineering. Its core is a tested and proven knowledge base that has been developed and continues to be updated frequently, through practices that have been documented, reviewed, and discussed by the software engineering community. Even more precisely, in this article, we refer to *dynamic testing* that comprises the activities that are performed to assess whether a software product works as expected when it is executed [34]. The ST taxonomy is shown in Figure 7, where, the white boxes represent ST domains and key concepts drawn from the SWEBOK [18], while the gray boxes are additional key concepts extracted from the analyzed secondary studies during the mapping process and missing in the SWEBOK. In the remainder of this section, we provide a short description for each domain and key concept of the taxonomy. Moreover, we indicated one or more references for the key concepts that are not described in the SWEBOK.

2.2.1 Test Target. The ST domain that defines the possible objects of the testing. The target can vary from a single module to an integration of such modules (related by purpose, use, behavior, or structure) and an entire system. In this domain, we recognized three relevant fields: (i) *Unit Testing*, which verifies the correct behavior, in isolation, of software elements that are separately testable; (ii) *Integration Testing*, which is intended to verify the correct interactions among software components; and (iii) *System Testing*, which is concerned with checking the expected behavior of an entire system.

2.2.2 Testing Objective. The ST domain defining the purpose of a testing process. Test cases can be designed to check that the functional specifications are correctly implemented. This objective is also defined in literature as conformance testing, correctness testing, functional testing, or feature testing [11]. However, in *Non-functional Testing*, several other nonfunctional properties may be verified as well, including *reliability*, *usability*, *safety*, and *security*, among many others quality characteristics such as *compatibility* [30] and *quality of service (QoS)* [1]. Other possible testing objectives are the following ones: (i) *Acceptance Testing*, which determines whether a system satisfies its acceptance criteria, usually by checking desired system behaviors against the customer's requirements; (ii) *Regression Testing*, which, according to [33], is *"...selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.."*; (iii) *Stress Testing*, which exercises the software at the maximum design load with the goal of determining the behavioral limits, and to test defense mechanisms in critical systems; (iv) *structural testing*, whose target is to cover the

internal structure of the system source code or model [89]; and (v) *GUI Testing*, which focuses on detecting faults related to the Graphical User Interface (GUI) and its code [89].

2.2.3 Testing Technique. The ST domain dealing with the detection of as many failures as possible. Testing techniques have the main goal of identifying inputs that produce representative program behaviors and assessing whether these behaviors are expected or not in comparison to specific oracles. Testing techniques have been classified on the basis of how they design or generate test cases. Possible testing techniques are described as follows: (i) *Combinatorial Testing*, where test cases are derived by combining interesting values for every pair of a set of input variables instead of considering all possible combinations; (ii) *Mutation Testing*, which uses mutants, i.e., mutated versions of the source code under test, as test goals to create or improve test suites; (iii) *Random Testing*, which generates tests purely at random; (iv) *Model-based Testing*, which is used to validate requirements, check their consistency, and generate test cases focused on the behavioral aspects of the software. The software under test is usually represented in a formal or semi-formal way by means of models; (v) *Equivalence Partitioning Testing*, which involves the partitioning of the input domain into a collection of subsets (or equivalent classes) based on a specified criterion or relation; (vi) *Requirement-based Testing*, which extracts test cases from requirements in any (partially) automated way [35]; (vii) *Concolic Testing* employs the symbolic execution of a program paired with its actual execution [11]; (viii) *Metamorphic-based Testing*, which uses metamorphic relationships for the test oracles definition [32]; (ix) *Concurrency Testing*, where tests are generated for verifying the behavior of concurrent systems [2]; and (x) *Statistical Testing*, where the test cases are generated starting from statistical models such as Markov Chains [12].

2.2.4 Testing Activity. The ST domain that outlines the activities that can be performed by testers and testing teams into well-defined controlled processes. Such activities vary from test planning to test output evaluation, in such a way as to provide assurance that the test objectives are met in a cost-effective way. Well-known testing activities presented in the literature are the ones presented in the following of this section. (i) *Test Case Generation* whose goal is to generate executable test cases based on the level of testing to be performed and the particular testing techniques. (ii) *Test Planning* is a fundamental activity of the ST process, its includes the coordination of personnel, availability of test facilities and equipment, creation and maintenance of all test-related documentation, and planning for the execution of other testing activities. (iii) *Test Logs Reporting* is used to identify when a test was conducted, who performed the test, what software configuration was used, and other relevant information to identify or reproduce unexpected or incorrect test results. (iv) *Defect Tracking* is the activity where the defects can be tracked and analyzed to determine when they were introduced into the software, why they were created (for example, poorly defined requirements, incorrect variable declaration, memory leak, programming syntax error), and when they could have been first observed in the software. (v) *Test Results Evaluation* is performed to determine whether the testing has been successfully executed. In most cases, "successful" means that the software performed as expected and did not have any major unexpected outcomes. Not all unexpected outcomes are necessarily faults, but are sometimes determined to be simply noise. Before a fault can be removed, an analysis and debugging effort is needed to isolate, identify, and describe it. (vi) *Test Execution* represents both the execution of test cases and the recording of the results of those test runs. Execution of tests should embody a basic principle of scientific experimentation: everything done during testing should be performed and documented clearly enough that another person could replicate the results. (vii) *Test Environment Development* regards the implementation of the environment that is used for testing. It should be guaranteed that the environment is compatible with the other software engineering tools adopted

during the testing process. It should facilitate the development and control of test cases, as well as the logging and recovery of expected results, scripts, and other testing materials. (viii) *Test Oracle Definition* is the activity performed either to generate automatically or to support the creation of test oracles [32]. (ix) *Test Case Design and Specification* is executed to design or to specify the testing cases. This activity usually starts from the analysis of the requirements of the system under test [30, 35]. (x) *Test Case Optimization/ Prioritization/Selection* is performed for the optimized reduction or prioritization or selection of test cases to be executed [43]. (xi) *Test Data Definition*, a.k.a. test data generation, is the activity where the data for test cases are produced [98]. (xii) *Test Repair* is, in essence, a maintenance activity. Within the course of this activity, test scripts are adjusted to changed conditions. The need for it lies in the fact that test scripts are fragile and vulnerable to the changes introduced by developers in a newer version of the tested software [98]. (xiii) *Flaky Test Prediction* is the activity where the tests expressing similar characteristics are identified and repaired. This activity significantly improves the overall stability and reliability of the tests [98]. A flaky test is considered as such when it reports false positive or false negative test result, or when adjustment was made to the test scripts and/or to the code of the system under test. (xiv) *Test Costs Estimation* has the main goal to predict the testing costs, mainly the testing time [30].

2.2.5 Software Testing Fundamentals. The ST domain that covers the *Testing Related Terminology*, such as basic definitions, basic terminology and key issues, and relationships between software testing and other software engineering activities.

3 TERTIARY SYSTEMATIC MAPPING PROTOCOL

In this section, we describe the research protocol adopted to conduct our tertiary systematic mapping study. The protocol was designed following the guidelines proposed by Petersen et al. [77] to fulfill the requirements of a structured process, whose execution details are provided in Section 4. Specifically, the protocol includes the following steps: (i) definition of goals and research questions, (ii) definition of the search string, (iii) selection of electronic databases, (iv) definition of inclusion and exclusion criteria, (v) definition of quality assessment criteria, and (vi) design of the data extraction form. We describe in detail each of these steps and their outcomes in the rest of this section.

3.1 Goal and Research Questions

The goal of our study is to understand how AI has been applied to support ST. To reach this goal, we defined nine research questions (RQs) grouped into two categories *publication space* and *research space* questions (as suggested by [77]). *Publication space* questions aim at characterizing the bibliographic information (i.e., venue, year of publication, authors' affiliation, etc.) of the identified sources (i.e., secondary studies). *Research space* questions aim at providing the answers needed to achieve the research goal.

***Publication Space (PS)** RQs.* We defined the following five publication space research questions:

PS-RQ1: How many secondary studies have been identified per publication year?
PS-RQ2: Which types of secondary studies have been executed?
PS-RQ3: What are the venues where the secondary studies have been published?
PS-RQ4: What are the authors' affiliation countries of the selected secondary studies?
PS-RQ5: What is the amount of primary studies analyzed by the selected secondary studies and how are they distributed over time?

Table 1. PICOC Main Terms and their Synonyms

View Point	Main Term	Synonyms
Population	Software Testing	Based Testing, Dynamic Testing, Static Testing, Test Oracle, Test Design, Test Execution, Test Report, Test Plan, Test Automation, Automated Test, Test Case, Bug Detection, Fault Detection, Error Detection, Failure Detection.
Intervention	Artificial Intelligence	AI, Linguistic, Computer Vision, Recommend System, Decision Support, Expert System, NLP, Natural Language Processing, Data Mining, Information Mining, Text mining, Learning, Supervised, Unsupervised, Rule-based, Training, Decision Tree, Neural Network, Bayesian network, Clustering, Genetic Programming, Genetic Algorithm, Evolutionary Programming, Evolutionary Algorithm, Evolutionary Computation, Ensemble Method, Search-based, Intelligent Agent, Naive Bayes, Ontology, Random Forest, Reasoning, SVM, Support Vector, Activation Function, Autoencoder, Backpropagation, Boosting, Cross-validation, Ground Truth, Ant Colony, Bee Colony, Particle Swarm, Robotics, Planning.
Comparison	N.A.	
Outcome	N.A.	
Context	Secondary Study	Survey, Mapping, Review, Literature Analysis

Research Space (RS) *RQs.* We defined the following four research space research questions:

RS-RQ1: What AI domains have been applied to support ST?
RS-RQ2: What domains of ST have been supported by AI?
RS-RQ3: Which ST domains have been supported by which AI domains, and how?
RS-RQ4: What are the future research directions of AI in ST?

3.2 Search String Definition

To systematically define the search string to be used for finding secondary studies of interest, we adopted the PICOC (Population, Intervention, Comparison, Outcome, and Context) criteria as suggested in Petersen et al. [77]. The main term of each of the PICOC view points are described in the following:

- *Population*: We identified *Software Testing* as the main term of this view point, since it is the domain of interest of our study.
- *Intervention*: We identified *Artificial Intelligence* as the main term of this view point, since our research questions are aimed at investigating how this science has been applied to the *population*.
- *Comparison*: This view point is not applicable in a systematic mapping study, since no effect of the *intervention* on the *population* can be expected.
- *Outcome*: this view point is not applicable in a systematic mapping study, since no effect of the *intervention* on the *population* can be expected.
- *Context*: We identified *Secondary Study* as main term of this view point, since it is the context where we expect to find sources.

To identify the keywords of the search string, we followed the general approach as suggested by Kitchenham and Charters [52]. Hence, we performed a break down of our research questions (see Section 3.1) into individual facets (one for each PICOC view point). Then, we generated a list of synonyms, abbreviations, and alternative spellings. Additional terms were obtained by considering subject headings used in journals and scientific databases. The main terms and the synonyms we

inferred for the PICOC view points are shown in Table 1. Finally, the search string was obtained by the conjunction *(AND)* of disjunction *(OR)* predicates, each built on the main term and the corresponding synonyms of a PICOC view point. Moreover, as suggested by the Kitchenham [52] and Petersen [77] guidelines, we checked our search string against four selected control papers (Garousi et al. [35], Trudova. et al. [98], Catal [22], and Durelli et al. [30]).[4] The resulting search string is shown in Box 1.

("Software Test*" **OR** "*Based Test*" **OR** "Dynamic Test*" **OR** "Static Test*" **OR** "Test* Oracle*" **OR** "Test* Design" **OR** "Test* Execution" **OR** "Test* Report*" **OR** "Test* Plan*" **OR** "Test* Autom*" **OR** "Autom* Test*" **OR** "Test Case*" **OR** "Bug Detection" **OR** "Fault Detection" **OR** "Error Detection" **OR** "Failure Detection") **AND** ((ai **OR** "Artificial Intelligence" **OR** linguistic **OR** "computer vision" **OR** "recommend* system*" **OR** "decision support" **OR** "expert system*" **OR** "NLP" **OR** "natural language processing" **OR** "data mining" **OR** "information mining" **OR** "text mining" **OR** "* learning") **OR** ("supervised" **OR** "unsupervised" **OR** "rule-based" **OR** "training" **OR** "decision tree*" **OR** "neural network*" **OR** "bayesian network*" **OR** "clustering" **OR** "genetic programming" **OR** "genetic algorithm*" **OR** "evolutionary programming" **OR** "evolutionary algorithm*" **OR** "evolutionary computation" **OR** "ensemble method*" **OR** "search-based" **OR** "intelligent agent*" **OR** "naive bayes" **OR** "ontolog*" **OR** "random forest*" **OR** "reasoning" **OR** "SVM*" **OR** "support vector" **OR** "activation function*" **OR** "autoencoder*" **OR** "backpropagation" **OR** "boosting" **OR** "cross-validation" **OR** "ground truth" **OR** "ant colony" **OR** "bee colony" **OR** "particle swarm" **OR** robotics **OR** planning)) **AND** (survey **OR** mapping **OR** review **OR** "secondary study" **OR** "literature analysis")

Box 1. Resulting search string.

3.3 Digital Libraries Selection

To retrieve candidate studies, we selected four of the most well-known digital libraries usually adopted for conducting literature review and mapping studies [4]. The digital libraries adopted in this study are: *ACM Digital Library*,[5] *IEEE Xplore*,[6] *Web of Science*,[7] and *Scopus*.[8] We adapted the search string to accommodate to the syntax required by each digital library search engine, hence we built four queries that apply our search string to the *title* and *abstract* attributes. Additionally, for *Scopus* and *Web of Science*, we limited the results to the *computer science* and *computer engineering* categories. Since the *ACM Digital Library* and the *IEEE Xplore* gather publications within *computer science* and *computer engineering* no restrictions have been applied in the corresponding queries.

3.4 Inclusion and Exclusion Criteria Definition

To support the selection of retrieved secondary studies, we defined exclusion and inclusion criteria. When defining these criteria, we acknowledged our complementary skills in AI and ST. Therefore, as we will mention in Section 4, we defined these criteria with the outright intention that its application would be supported by classifiers with the required skills to properly apply the criteria in the context of the expertise of each of our fields.

Exclusion Criteria (EC). We excluded a publication if at least one of the following 6 EC applies:

(EC1) The study focuses on the testing of AI-based software systems rather than the application of AI to ST.

[4]Control papers are used to calibrate the search string by representing, to the best of the research team's knowledge, the characteristics of the "ideal" type of research publication that the team is looking for. Both the Kitchenham [52] and Petersen [77] guidelines highlight the need to check trial search strings against a list of already known studies. We used the Scopus digital library for the search string validation.

[5]http://dl.acm.org/

[6]http://ieeexplore.ieee.org/

[7]https://www.webofknowledge.com/

[8]http://www.scopus.com/

Table 2. Quality Assessment Criteria

Criteria	Yes *(+1.0)*	Partly *(+0.5)*	No *(+0.0)*
QC1: were there explicit research questions?	Source presents the research questions, and these guide the secondary study through the application of PICOC (or a variation)	Source present the research questions, and it guides the secondary study without a formal mapping to the search strategy	Source does not present research questions that guide the secondary study
QC2: were inclusion and exclusion criteria reported?	Inclusion and exclusion criteria are explicitly defined	Implicit inclusion/exclusion criteria	Inclusion and exclusion criteria are not defined and cannot be inferred
QC3: was the search strategy adequate?	Searched in 4 or more digital libraries and included additional search strategies	Searched in 3 or 4 digital libraries with no extra search strategies	Searched up to 2 digital libraries
QC4: was the quality of the included studies assessed?	Quality criteria explicitly defined and extracted for each secondary study	Quality issues of primary studies addressed by research questions	No explicit quality assessment
QC5: were there sufficient details about the individual included studies presented?	Each primary study can clearly be traced from the information provided.	Only summary information is provided for each individual study.	Results of individual studies are neither specified nor summarized.
QC6: were the included studies synthesized?	Data was extracted, summarized and interpreted.	Data was extracted and summarized but not interpreted.	Data was not summarized nor interpreted.

(EC2) The study focuses on the application of AI for either the prediction or the analysis, or the localization of: (i) errors; (ii) faults; (iii) bugs; or (iv) failures.
(EC3) The study is a duplicate of another candidate paper.
(EC4) The study does not provide substantially different contribution compared to another candidate work written by the same authors.
(EC5) The study has another candidate paper, written by the same authors, which is an extended version of it.
(EC6) The study is a tertiary systematic mapping.

We remark that, to apply EC2, we took special attention to confirm that the source shares our focus on dynamic testing. In particular, we are looking to exclude studies (systematic mapping or reviews) that are not focused on the design and execution of test cases.

Inclusion Criteria (IC). We included a publication i.i.f. all the following 4 IC apply:

(IC1) The study is a secondary study.
(IC2) The study addresses the topic of AI applied to ST.
(IC3) The study is a peer-reviewed paper.
(IC4) The study is written in English.

3.5 Quality Assessment Criteria Definition (QC)

To filter-out low quality publications, we scored each candidate paper according to a list of six quality assessment criteria inspired by Kitchenham et al. [53]. We report in Table 2 such QCs along with the rationale we adopted to assign a score $\in \{0.0, 0.5, 1.0\}$ to each paper. We evaluate the overall quality of a candidate by summing up the six QC scores and excluding those papers reaching an overall score lower than 3.0.

3.6 Data Extraction Form Design

To support the data extraction process, we designed the data extraction form reported in Table 3. This form was used to report the pieces of evidence—extracted from the selected papers—that will be analyzed to answer the RQs.

Table 3. Data Extraction form

Publication Space			
ID	*Field*	*Description of the extracted data*	*RQ*
1	Title	Title of the secondary study	–
2	Abstract	Abstract of the secondary study	–
3	Authors	Authors list of the secondary study	–
4	Year	Publication year of the secondary study	PS-RQ1
5	Study Type	Type of secondary study, i.e., SM, Review, SLR, Multivocal	PS-RQ2
6	Venue	Name of the venue where the secondary study was published	PS-RQ3
7	Venue Type	Type of the venue where the secondary study was published	PS-RQ3
8	Institutions	Authors' Institutions list of the secondary study	PS-RQ4
9	Primary Studies	List of primary studies reviewed by the secondary study	PS-RQ5
Research Space			
ID	*Field*	*Description of the extracted data*	*RQ*
10	AI Space	List of extracted sentences on AI domain concepts	RS-RQ1
11	ST Space	List of extracted sentences on ST domain concepts	RS-RQ2
12	AI applied to ST Space	List of extracted sentences on AI applied to ST	RS-RQ3
13	Future Directions Space	List of extracted sentences on future directions in AI applied to ST	RS-RQ4

In this table, we enumerate and describe the fields composing the data extraction forms for the PS and RS RQs.

The form includes a list of fields organized in two sections, one dedicated to the publication space RQs and the other dedicated to the research space RQs. For each field, we provide a name, a brief description of the data that the field is meant to collect, and the RQ for which the field is used for.

4 TERTIARY SYSTEMATIC MAPPING EXECUTION

In this section, we describe the execution of the tertiary systematic mapping study we conducted with the protocol that we introduced in Section 3. Specifically, in Section 4.1, we provide details about the selection process while, in Section 4.2, we provide details about the data extraction process.

4.1 Selection Process Execution

The process followed to select secondary studies is shown in Figure 1. The figure provides a representation of the executed steps and their outcomes. The selection process is based on the execution of the two stages described in the following. The full selection process was executed on June 2021 and repeated on May 2022 to ensure that we did not miss any recent secondary study on the investigated topic.

4.1.1 First Stage. This stage was executed to select a preliminary set of secondary studies and relies on the sequential execution of the following four steps:

(1) *Secondary studies retrieval from the digital libraries*: In this step, the queries (introduced in Section 3.3) were submitted to the four digital libraries reported in Section 3.3.[9] As a result, 877 secondary studies were retrieved.
(2) *EC and IC application to title, abstract, and keywords*: The 877 papers were divided into two groups. The title, abstract, and keywords of the secondary studies in each group were analyzed by two researchers—one AI expert and one ST specialist—to apply IC and EC presented in Section 3.4. At the end of this step, 806 studies were excluded, since both researchers

[9]No time limits were considered in the search.

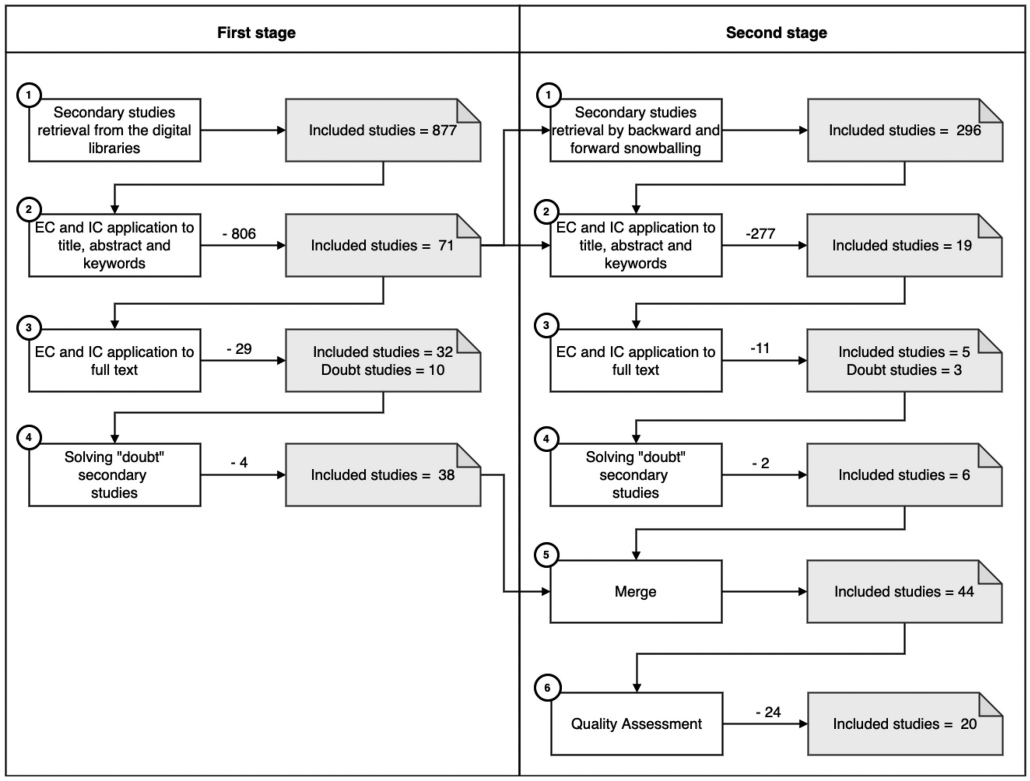

Fig. 1. Diagram of the secondary studies selection process execution.

reached the same consensus to remove them. The remaining 71 papers were included and passed to the next step.

(3) *EC and IC application to full text*: The 71 papers were divided into two groups and the full text of each paper was read by two researchers—an AI expert and an ST specialist—to apply again the IC and EC. In the end, 29 studies were excluded, 32 were included, and 10 papers were labeled as a "doubt." All doubts came from studies for which no agreement was reached.

(4) *Dealing with secondary studies classified "doubt"*: Two additional researchers—one AI expert and one ST specialist—were involved to reach an agreement on "doubt" papers. To this aim, all four researchers performed an additional discussion based on the papers' full read and analysis. At the end of the discussion, 4 studies were excluded whereas the remaining 6 were selected. As a consequence, the final set of selected papers included 38 studies.

4.1.2 Second Stage. This stage refers to a snowballing process [104] that was conducted as a complementary search strategy to mitigate the threat of missing literature. As shown in Figure 1 the stage relies on the execution of six sequential steps, three of which (i.e., steps 2, 3, and 4) involve the same steps that we described in the first stage.

(1) *Secondary studies retrieval by backward and forward snowballing*: In this automatic step, 296 secondary studies were retrieved by applying backward and forward snowballing to the 63 papers selected in the *First selection (title, abstract and keywords)* step of the First stage.

Specifically, for the backward snowballing, we collected all studies cited by the 63 papers using their references. For the forward snowballing, we used Google Scholar[10] instead of the four digital libraries already exploited in the first stage, as it allowed us to fully automate the retrieval of papers citing one or more of the 63 papers.

(2) *EC and IC application to title, abstract and keywords*: As a result of this step, 277 secondary studies were excluded by applying IC and EC. We remark that, to apply the EC3, we needed, as input to this step, the 63 papers selected in the *EC and IC application to title, abstract and keywords)* step of the First stage of the process. As a result, 19 papers were included for full reading in the following selection steps.

(3) *EC and IC application to full text*: In this step, among the 19 secondary studies, 11 were excluded, five were included, and the remaining three papers were labeled as "doubt" and needed an extra analysis.

(4) *Solving "doubt" secondary studies*: Among the three papers labeled as "doubt," two were excluded, and the other one was included. As a consequence, six secondary studies were obtained by snowballing.

(5) *Merge*: The papers selected in the two stages were merged to define a set of 44 candidate secondary studies.

(6) *Quality Assessment*: In this step, each paper was analyzed by one of the researchers who scored the source according to the six quality criteria described in Section 3.4. The studies with a quality score lower than 3 were excluded. Borderline papers, i.e., papers with a quality score between 2 and 4, were discussed by all the researchers. As a result of the quality assessment (see Table 4), 20 secondary studies were included into the final set of selected papers.

Hereafter, we refer to each selected paper following the identifier (i.e., F1, F2, ..., F20) provided in Table 4 at column *ID*.

4.2 Data Extraction Execution

Since our RQs (see Section 3.1) cross two different research areas, we divided the authors into two groups, each containing an AI expert and an ST specialist. The two members of each group collaborated in the extraction of the pieces of evidence from each of the 20 selected studies using the extraction form shown in Table 3. Finally, to reach a large consensus, the two groups shared the extracted pieces of evidence and discussed the differences.

5 DATA ANALYSIS

In this section, we describe the results of the analysis performed on the extracted data (see Section 4.2) to answer our RQs (see Section 3.1). Specifically, in Section 5.1, we provide the answers to our PS-RQs, while in Section 5.2 to our RS-RQs.

5.1 Publication Space Research Questions—Results

In the following sections, we answer the five PS-RQs of this study.

5.1.1 PS-RQ1. How many secondary studies have been identified per publication year? To reply to the first publication space question, we depicted in Figure 2 the distribution of selected secondary studies per publication year. As shown in Figure 2, 2009 is the first year for which we selected a secondary study. Most of the selected secondary studies (75%) were published in the past six

[10] https://scholar.google.com/

Table 4. Quality Assessment Results

ID	Secondary Study Title	QC1	QC2	QC3	QC4	QC5	QC6	QS
F1	A systematic mapping addressing Hyper-Heuristics within Search-based Software Testing [11]	1	1	1	0.5	1	1	5.5
F2	NLP-assisted software testing: A systematic mapping of the literature [35]	1	1	1	0.5	1	1	5.5
F3	Analyzing and documenting the systematic review results of software testing ontologies [96]	1	1	1	1	1	1	6
F4	A systematic literature review on semantic web enabled software testing [25]	1	1	1	1	1	1	6
F5	Artificial intelligence in software test automation: A systematic literature review [98]	1	1	1	1	1	1	6
F6	On the application of genetic algorithms for test case prioritization: A systematic literature review [22]	0.5	1	1	0.5	0.5	0.5	4
F7	A systematic review of search-based testing for non-functional system properties [1]	1	0.5	1	0.5	1	1	5
F8	Systematic Literature Review on Search-based mutation testing [43]	1	0	1	0	0.5	0.5	3
F9	The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review [89]	0.5	1	1	0	0.5	1	4
F10	A systematic review on search-based mutation testing [94]	1	1	1	0	1	1	5
F11	A systematic review of the application and empirical investigation of search-based test case generation [3]	1	1	1	0.5	0.5	1	5
F12	Machine learning applied to software testing: A systematic mapping study [30]	0.5	1	1	0	1	0.5	4
F13	Using Genetic Algorithms in Test Data Generation: A Critical Systematic Mapping [81]	0.5	1	1	0	1	1	4.5
F14	Ontologies in software testing: A Systematic Literature Review [28]	0.5	1	1	0	1	1	4.5
F15	A comprehensive investigation of natural language processing techniques and tools to generate automated test cases [2]	0.5	1	0.5	1	1	1	5
F16	Search-based Higher Order Mutation Testing: A Mapping Study [57]	1	1	0.5	0.5	0	0	3
F17	Trend Application of Machine Learning in Test Case Prioritization: A Review on Techniques [50]	1	1	0	1	0.5	0.5	4
F18	Using machine learning to generate test oracles: A systematic literature review [32]	1	0	0.5	0	0.5	1	3
F19	Test case selection and prioritization using machine learning: a systematic literature review [74]	1	1	1	0	1	1	5
F20	A Systematic Literature Review on prioritizing software test cases using Markov chains [12]	1	1	1	0	1	1	5
	A survey on regression testing using nature-inspired approaches [9]	0	0	0	0	0.5	0.5	1
	The Generation of Optimized Test Data: Preliminary Analysis of a Systematic Mapping Study [100]	0.5	0	0.5	0	0.5	0.5	2
	Artificial Intelligence Applied to Software Testing: A Literature Review [58]	0	0	0.5	0	0.5	0.5	1,5
	Use of Evolutionary Algorithm in Regression Test Case Prioritization: A Review [76]	0.5	0	0	0	0.5	0	1
	An extensive evaluation of search-based software testing: a review [48]	0.5	0	1	0	0	0.5	2
	Integration of properties of virtual reality, artificial neural networks, and artificial intelligence in the automation of software tests: A review [91]	0.5	0	0	0	0.5	0.5	1,5
	A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms [10]	0	0	0	0	0.5	0	0.5
	A critical review on automated test case generation for conducting combinatorial testing using particle swarm optimization [79]	0	0	0	0	1	1	2
	A systematic review of software testing using evolutionary techniques [68]	0	0	0	0	1	1	2
	Evolutionary algorithms for path coverage test data generation and optimization: A review [67]	0	0	0	0	1	1	2
	Search-based secure software testing: A survey [49]	1	0	0	0	0	0	1
	Multi-objective test case minimization using evolutionary algorithms: A review [101]	0	0	0	0	1	1	2
	Literature survey of Ant Colony Optimization in software testing [95]	0	0	0	0	0	0.5	0.5
	Heuristic search-based approach for automated test data generation: A survey [60]	0.5	0	1	0	0.5	0.5	2,5
	Soft computing-based software test cases optimization: A survey [92]	0	0	0	0	0.5	0	0.5
	Bayesian concepts in software testing: An initial review [82]	0.5	0.5	1	0	0	0.5	2,5
	Search-based techniques and mutation analysis in automatic test case generation: A survey [26]	0	0	0	0	1	0.5	1,5
	A Survey on Testing software through genetic algorithm [23]	0	0	0	0	1	1	2
	Evolutionary software engineering, a review [63]	0	0	0	0	1	0.5	1,5
	Search-based software test data generation: A survey [65]	0	0	0	0	0.5	0.5	1
	Nature-inspired approaches to test suite minimization for regression testing [8]	1	0	0	0	0.5	0	1,5
	Review of search-based techniques in software testing [83]	0	0	0	0	0	0	0
	Object-Oriented Evolutionary Testing: A Review of Evolutionary Approaches to the Generation of Test Data for Object-Oriented Software [71]	0	0	0	0	0	0	0
	A systematic review of agent-based test case generation for regression testing [5]	1	0.5	0.5	0	0	0	2

For each paper, we report the corresponding quality scores.

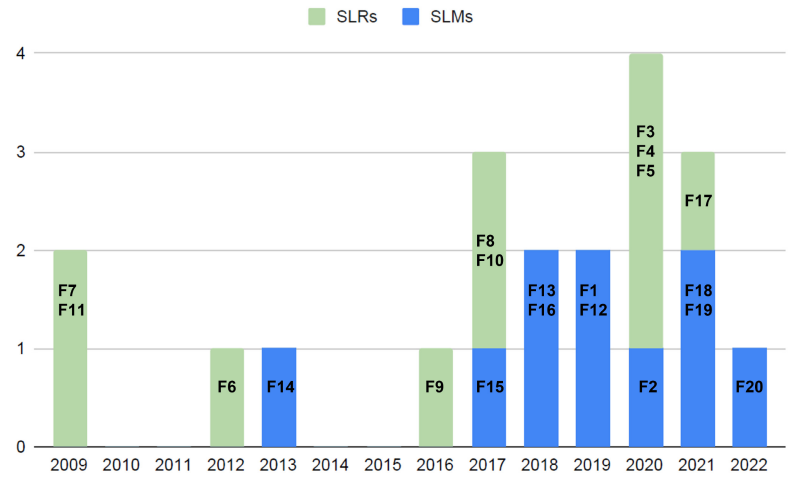

Fig. 2. Distribution of secondary studies per publication year and type.

years (2017 to 2022), thus showing an increasing interest in the research community in conducting secondary studies about the application of AI in ST.

5.1.2 PS-RQ2. Which types of secondary studies have been executed? The second PS-RQ is about the types of selected secondary studies. For each study, we report in Figure 2 the corresponding type, i.e., either *Systematic Literature Review (SLR)* in light green color or *Systematic Literature Mapping (SLM)* in blue color. We followed the guidelines defined by Kitchenham and Charters [52] to verify whether a secondary study was correctly classified as SLR or SLM by its authors, and changed the classification when needed. For instance, F14 and F15 were originally classified as SLRs by their authors. After a careful analysis, we opted to classify them as SLMs, since the authors: (i) did not perform a quality assessment of selected primary studies; (ii) summarized the selected works without executing a meta-analysis. From the data presented in Figure 2, we can observe that our selection includes 10 (50%) SLRs and 10 (50%) SLMs.

5.1.3 PS-RQ3. What are the venues where the secondary studies have been published? The third PS-RQ aims to analyze the venues where the selected secondary studies have been published. Table 5 reports the type, name, and rank (*Scimago Journal & Country Rank*—SJR quartile[11] for journal papers and *Computing Research and Education*—CORE rank[12] for conference papers) of the venues of the 20 selected secondary studies. The table shows that 14 (70%) studies were published in journals and the remaining 6 studies (30%) were part of the proceedings of conferences, workshops, symposiums, or seminars. It is worth observing that 13 of the 14 journal papers have been published in top-ranked venues (according to the SJR quartile in which the venue is classified), with 6 of them published in the *Information and Software Technology Journal*.[13] Thus, from our selection, we can derive that the topic of AI applied to ST is largely covered by top-ranked journals.

5.1.4 PS-RQ4. What are the authors' affiliation countries of the selected secondary studies? The fourth PS-RQ is aimed at analyzing the countries of affiliation of the authors of the selected studies.

[11] https://www.scimagojr.com/
[12] http://portal.core.edu.au/conf-ranks/
[13] https://www.journals.elsevier.com/information-and-software-technology

Table 5. Secondary Studies Per Venues' Types and Names

Venue Type	Venue Name	SJR Quartile	CORE Rank	Study ID
Journal	Information and Software Technology	Q1		F1, F2, F3, F7, F10, F20
	ACM Computing Surveys	Q1		F13
	Applied Soft Computing	Q1		F9
	e-Informatica Software Engineering Journal	Q3		F8
	Empirical Software Engineering	Q1		F19
	IEEE Access	Q1		F17
	IEEE Transactions on Reliability	Q1		F12
	IEEE Transactions on Software Engineering	Q1		F11
	Journal of Systems and Software	Q1		F4
Conference	Brazilian Symposium on Systematic and Automated Software Testing		Not Available	F16
	International Conference on Evaluation of Novel Approaches to Software Engineering		B	F5
	International Conference on Internet of things, Data and Cloud Computing		C	F15
	International Workshop on Evidential Assessment of Software Technologies		Not Available	F6
	International Workshop on Test Oracles		Not Available	F18
	Seminar on Ontology Research in Brazil		Not Available	F14

Among the 20 selected studies, we found 68 different authors, with 5 authors (i.e., Érica Ferreira de Souza, Juliana Marino Balera, Lionel C. Briand, Nandamudi Lankalapalli Vijaykumar, and Juliana Marino Balera) involved in two studies each. We analyzed the countries of affiliations of these 68 authors, resulting in 16 unique affiliation countries. Since three authors reported a second affiliation country and five authors were involved in two different studies, we counted a total of 76 affiliations. Most of the selected studies have authors with affiliations from only one country, except studies F1 [11] (Brazil and UK), F2 [35] (Austria and Northern Ireland), and F3 [96] (Argentina and Uruguay), which included authors with affiliations from two different countries each. Figure 3 shows a world map of the authors' affiliation countries, with each color representing a different value for the number of affiliations. In particular, 27 (35.53%) affiliations are counted for *Brazil*, 10 (13.16%) for *Malaysia*, 6 (7.89%) for *Sweden*, 4 (5.26%) for Argentina, Canada, Norway, and Pakistan, each, 3 (3.95%) for the Czech Republic, India, and Iran each, 2 (2.63%) for Austria, United Kingdom, and Uruguay each, and 1 (1.32%) for Luxembourg and Turkey, each. From the extracted data, we can observe that most of the affiliations (33 over 76) are located in South America (Brazil, Argentina, and Uruguay). Interestingly, affiliation countries that typically dominate in computer science or computer engineering (e.g., USA and China) do not occur in our observations.[14]

5.1.5 PS-RQ5. What is the amount of primary studies analyzed by the selected secondary studies, and how are they distributed over time? The goal of this RQ is twofold: (i) to compute the number of primary studies that have been reviewed by the selected secondary studies and (ii) to understand how these primary studies are distributed over the publication years. Figure 4 shows, for each selected secondary study, the number of reviewed primary studies and how many of these studies are unique, i.e., works that have not been reviewed by any other secondary study. Looking at the figure, it shows that the 20 selected secondary studies analyzed a total of 807 primary studies, of which 710 (87.98%) were unique. Figure 5 shows the distribution of the unique primary studies per publication year. The figure shows that the reviewed 710 unique primary studies cover a period of 27 years, going from 1995 to 2021; 444 (62.5%) of these studies have been published in the past 10 years and 264 (37.18%) from 2015 to 2021. Primary studies that were "unique" in any of the older secondary studies could, in theory, have been found by the newer secondary studies. However, the publication date of a primary study would be just one of the factors that may lead it to be included in only one of the secondary studies. In addition, the search protocols for secondary studies also vary greatly in the choice of search strings and inclusion and exclusion criteria, leading to the great diversity of primary studies selected. The large amount of unique primary studies reviewed

[14] https://www.natureindex.com/annual-tables/2021/country/all

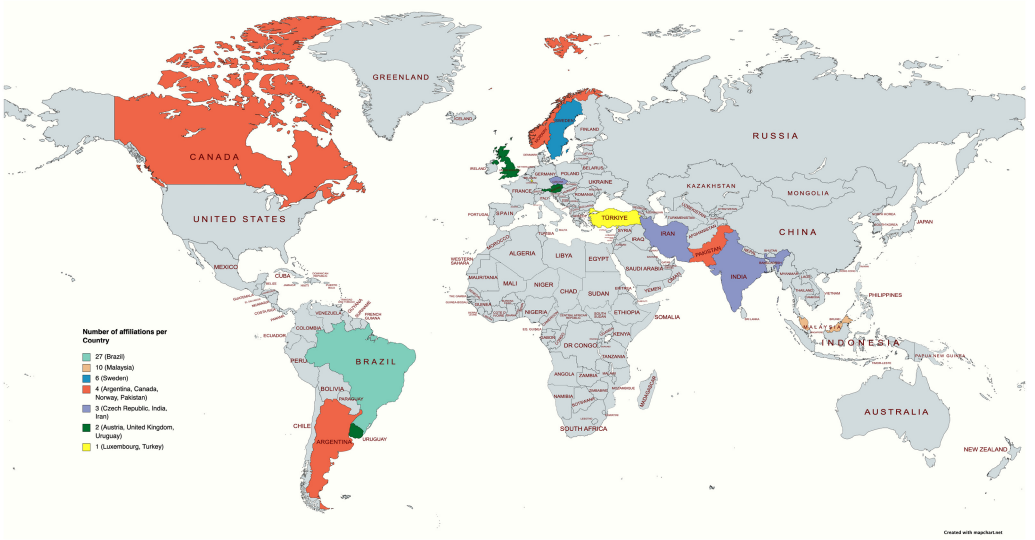

Fig. 3. World map of the authors' affiliation countries.

by the selected secondary studies[15] and their distribution over time leads us to two interesting observations. First, we can state that our set of secondary studies is representative of the research conducted on the topic of AI applied to ST. Moreover, as will be confirmed by RS-RQ1 and RS-RQ2 (see Sections 5.2.1 and 5.2.2) the set of unique primary studies reviewed by these works is a significant sample of primary studies covering broad aspects of AI in ST. Second, we can infer that the topic of AI applied to ST is of interest to the research community and that the interest has grown over the past decade. Finally, it is worth observing that the research topic of AI applied to ST is not new for the research community, indeed the first 19 (2.67%) primary studies in this field date from the late 1990s.

5.2 Research Space Research Questions—Results

In the following sections, we answer the four RS-RQs of our study.

5.2.1 RS-RQ1. What AI domains have been applied to support ST? To identify the AI domains from which solutions were applied to support ST, we analyzed the list of sentences about the applied AI domains that were extracted from our sources during the data extraction process with reference to the taxonomy introduced in Section 2.1. As a result of this analysis, in Figure 6, we report, for each AI domain concept, the list of secondary studies in which we found evidence of its application in ST. The most important findings we can derive from Figure 6 follow: (1) most of the secondary studies (11 of 20) investigated the application of AI solutions (i.e., algorithms, models, methods, techniques, etc.) belonging to the *Planning and Scheduling / Searching / Optimization* sub-domains to ST. Specifically, most surveyed AI solutions belonging to this domain are: *evolutionary algorithms*, *genetic algorithms*, and *metaheuristic optimisation*; (2) 9 of 20 secondary studies focused on the application of *Machine learning* solutions. In particular, F12 [30] and F5 [98] covered almost all the concepts in this AI domain; (3) 6 of 20 secondary studies surveyed the support provided by

[15]We can assume that 710 is only a part of the primary works in the literature on the subject, and thus that the number of published primary studies is even higher.

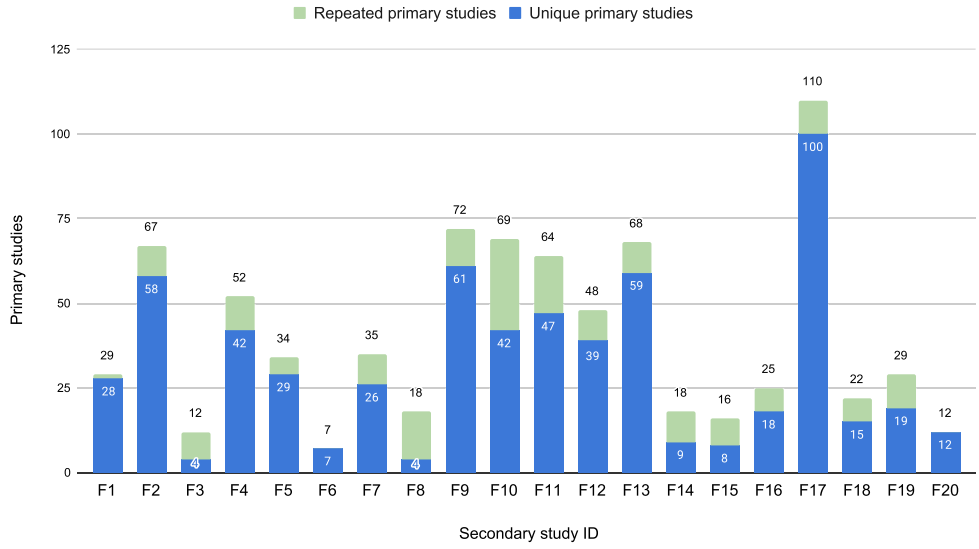

Fig. 4. Primary studies selected by each secondary study (including repeated primary studies).

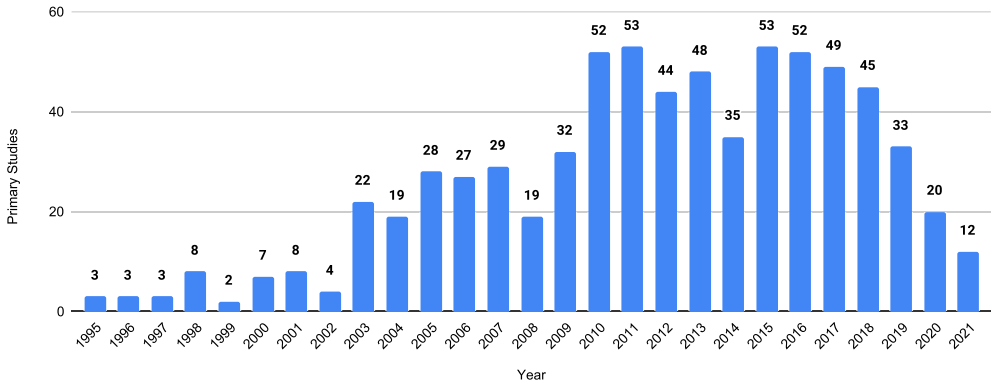

Fig. 5. Distribution of unique primary studies per publication year.

Knowledge representation/Automated reasoning/Common sense reasoning AI solutions. Specifically, most of these studies analyzed the use of ontologies and F4 [25] is the study that surveyed the use of most of the concepts in this AI domain; (4) few secondary studies surveyed works on the application of *Natural language processing* (5 of 20), *Multi-agents systems* (3 of 20), and *Computer vision* (1 of 20). It is worth noticing that, within the *Natural language processing* domain, the most surveyed applications are based on *text mining*, while only one study surveyed applications of *word embeddings*. Notably, only one of the secondary studies analyzed the use of *image processing* techniques belonging to *Computer vision*.

By analyzing the publication years of the selected studies (see Figure 2), we can observe that most of the works (6 of 9) surveying the application of *machine learning* to ST have been published very recently (2020 or later). This indicates a growing interest in this AI domain. Similarly, 4 of 5 studies investigating the use of *Natural language processing* have been published after 2020, highlighting the timeliness of this research field, with a special focus on *text mining* and *word embedding*.

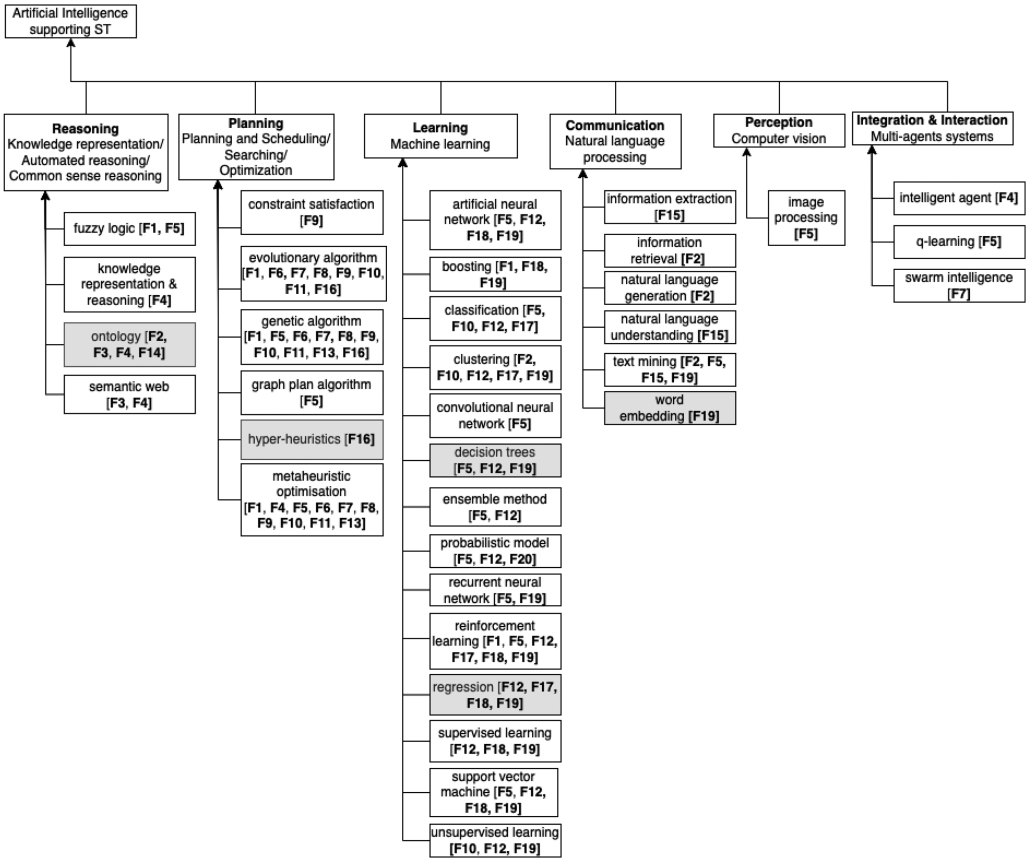

Fig. 6. The resulting excerpt taxonomy of AI supporting ST, built starting from the EU *AI Watch* report. Gray boxes represent key concepts not explicitly included in the *AI Watch* [88] report and added as a result of the data analysis process. Each concept is annotated with the labels of secondary studies in which it was surveyed. Original domain labels are reported in bold inside sub-domains boxes.

Secondary studies analyzing the use of *Planning and Scheduling/Searching/Optimization* cover a period spanning from 2009 to 2020, showing a consolidated and still of interest research topic.

5.2.2 RS-RQ2. What domains of ST have been supported by AI? Similar to what we did to answer RS-RQ1, we analyzed the sentences collected from each secondary study during the data extraction process and annotated each study with the ST domain concepts involved in them, according to the taxonomy introduced in Section 2.2. The result of this analysis is shown by Figure 7, where, for each ST domain, we report the list of secondary studies in which we found pieces of evidence of the application of an AI solution to the specific ST domain. From Figure 7, we can observe the following: (1) Almost all selected secondary studies (19 of 20) have surveyed studies about the application of AI to the *Testing activity* ST domain. In particular, the most recurrent ST concepts of this domain are: *Test Case Optimization/Prioritization/Selection* (11 of 20), *Test Data Definition* (10 of 20), and *Test case generation* (8 of 20); (2) 12 secondary studies surveyed the use of AI in the *Testing Technique* domain. In this ST domain, *Mutation Testing* and *Requirement-based Testing* are the testing techniques for which more secondary studies found evidence of AI applications

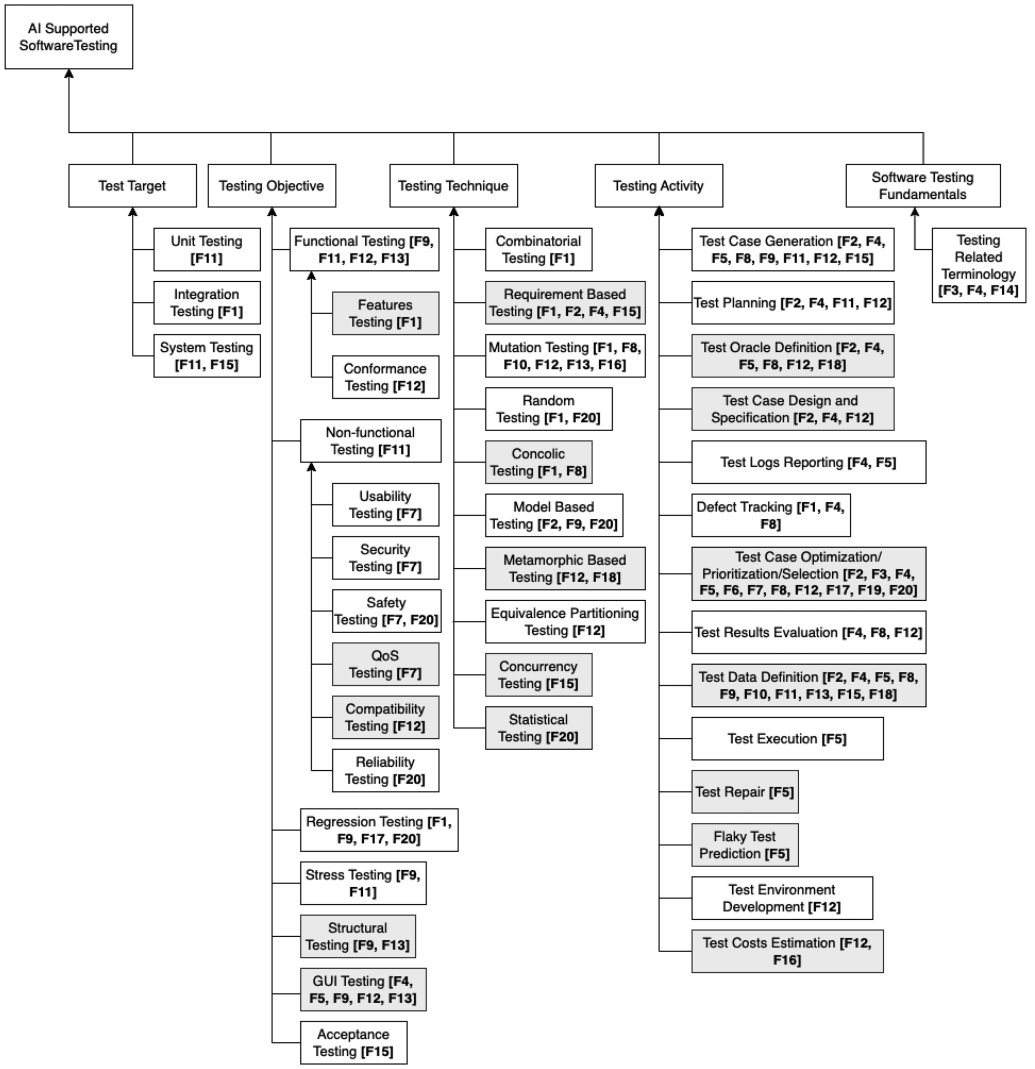

Fig. 7. The resulting excerpt AI supported ST taxonomy, built starting from the SWEBOK [18]. Gray boxes represent key concepts not explicitly included in the SWEBOK and added as a result of the data analysis process. Each concept is annotated with the labels of secondary studies in which it was surveyed.

(6 and 4 studies, respectively); (3) 11 secondary studies showed evidences of AI applied in the *Testing Objective* ST domain. In particular, 5 studies showed the use of AI to support *Functional Testing*, 5 studies analyzed primary sources where AI was applied to *Non-functional Testing*, 5 studies showed evidence on the application of AI to *GUI Testing*, and 4 surveyed the use of AI for *Regression Testing*; (4) few secondary studies (3 of 20), reported evidence on the use of AI in the *Test Target* ST domain. Among these 3 secondary studies, 1 covered the use of AI in *Unit Testing*, 1 the application of AI in *Integration Testing*, and 2 the AI support in *System Testing*; (5) *Software Testing Fundamentals* ST domain has been covered by 3 secondary studies. All these works reported evidence on the use of AI to support the introduction and standardization of terms and definitions in the field of the *Testing Related Terminology*.

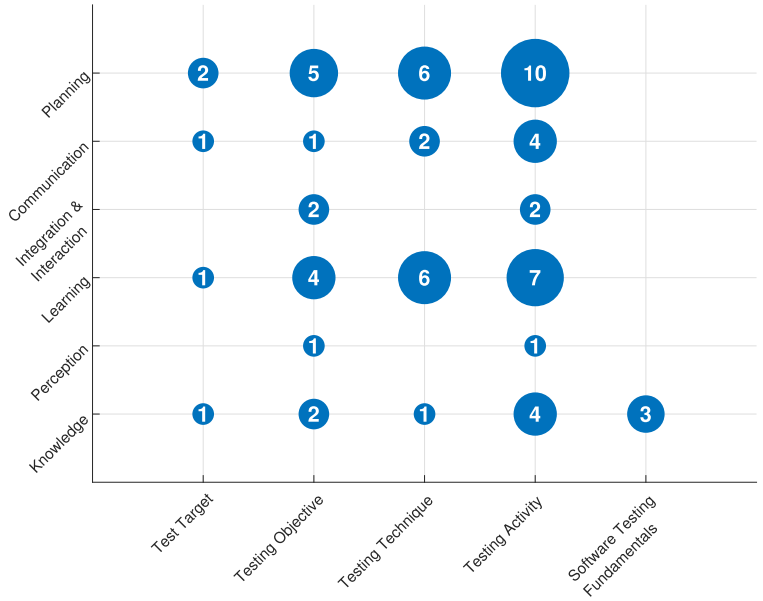

Fig. 8. AI and ST domain pairs covered by the selected secondary studies. For each pair of domains, we report the count of distinct secondary studies surveying the corresponding applications of AI to ST.

Overall, the most important finding we can derive from Figure 7 is the evidence of an intense application of AI to: (1) the development of test cases, including their generation and the definition of test cases' input and expected output, i.e., test oracles. To aid the test oracle definition, AI has been applied to metamorphic-based testing and to GUI testing; (2) the management of the test cases, particularly, their prioritization and selection, which is confirmed by the use of AI for regression testing; (3) the generation of test cases from requirements using natural language processing and knowledge representation techniques; (4) the detection of equivalent mutants and the generation of new mutants in mutation testing techniques; and (5) the testing of both functional and non-functional requirements.

5.2.3 RS-RQ3. Which ST domains have been supported by which AI domains and how? To answer RS-RQ3, we discuss the evidence collected from the selected secondary studies concerning what AI domains have been applied to support what ST domains. The bubble chart in Figure 8 reports the number of secondary studies that investigated the application of a given AI domain to a specific ST domain. From the chart, we can derive the following observations: (1) *Testing Activity* and *Testing Objective* are the only two ST domains for which we found evidence of the application of solutions from all the AI domains. Also, with the exception of *Software Testing Fundamentals*, the AI domains *Planning*, *Communication*, *Learning*, and *Knowledge* have been applied to all ST domains; (2) *Knowledge* is the only AI domain for which we found evidence of applications to *Software Testing Fundamentals*, moreover, it is the only AI domain involved in all the ST areas; (3) the majority of selected secondary studies (10 of 20) analyzed the application of AI techniques belonging to the *Planning* domain for supporting the *Testing Activity*, thus being the most surveyed interplay of AI and ST; (4) the second most surveyed interplay of AI and ST is *Learning* applied to *Testing Activity*. Moreover, we evidenced that *machine learning* is the only key concept belonging to the *Learning* AI domain that has been exploited in this ST domain; (5) very few secondary studies surveyed the application of the *Integration & Interaction* and the *Perception AI domains* to ST. More

Table 6. (a) First Part of the Resulting Mapping

		Knowledge				Planning						Communication					
		fuzzy logic	*knowledge representation and reasoning*	*ontology*	*semantic web*	*constraint satisfaction*	*evolutionary algorithm*	*genetic algorithm*	*graph plan algorithm*	*hyper-heuristics*	*metaheuristic optimisation*	*information extraction*	*information retrieval*	*natural language generation*	*natural language understanding*	*text mining*	*word embedding*
Test Target	*Unit Testing*						F11	F11			F11						
	Integration Testing	F1					F1	F1			F1						
	System Testing						F11	F11			F11	F15			F15	F15	
Testing Objective	*Functional Testing*					F9	F9	F9, F13			F9, F11, F13						
	Features Testing						F1	F1			F1						
	Conformance Testing																
	Non-functional Testing							F11									
	Usability Testing						F7	F7			F7						
	Security Testing						F7	F7			F7						
	Safety Testing						F7	F7									
	QoS Testing							F7									
	Compatibility Testing																
	Reliability Testing																
	Regression Testing						F1	F1, F9			F1, F9						
	Stress Testing							F9, F11			F9						
	Structural Testing					F9		F9, F13			F9						
	GUI Testing	F5	F4					F9, F13			F9						
	Acceptance Testing											F15			F15	F15	
Testing Technique	*Combinatorial Testing*						F1	F1			F1						
	Requirement-based Testing		F4				F1	F1			F1	F15	F2		F15	F2,F15	
	Mutation Testing						F1, F8, F10, F16	F1, F8, F10, F13		F16	F1, F8, F10						
	Random Testing						F1	F1			F1						
	Concolic Testing						F1, F8	F1, F8			F1, F8						
	Model-based Testing					F9										F2	
	Metamorphic-based Testing																
	Equivalence Partitioning																
	Concurrency Testing															F15	
	Statistical Testing																
Testing Activity	*Test Case Generation*		F4	F2, F4	F4		F9, F11	F5, F8, F9, F11			F5, F8, F11	F15			F15	F2,F5, F15	
	Test Planning		F4	F4	F4			F11									
	Test Oracle Definition		F4	F4	F4			F8	F5							F2	
	Test Case Design and Specification			F4	F4											F2	
	Test Logs Reporting			F4	F4												
	Defect Tracking			F4	F4		F1	F1, F8			F1						
	Test Case Optimization /Prioritization/Selection		F4	F3, F4	F3, F4		F6	F6, F7			F4, F5, F6, F7					F2, F19	F19
	Test Results Evaluation			F4	F4			F8									
	Test Data Definition			F4	F4	F9	F8	F5, F8, F10, F11			F8, F11, F13			F2		F15	
	Test Execution	F5															
	Test Repair																
	Flaky Test Prediction																
	Test Environment Development																
	Test Costs Estimation							F16									
Software Testing Fundamentals	*Testing Related Terminology*		F4	F3, F4, F14	F3, F4												

Rows and columns represent ST and AI concepts, respectively. Cells include the selected secondary studies from which we extracted the evidence of applications.

precisely, *Multi-agent systems* and *Computer vision* are the only AI key concepts belonging to these domains for which we had evidence of application in ST; (6) the *Software Testing Fundamentals* domain characterizes the *Software Testing Fundamental* terms and definitions. This justifies why *Knowledge* is the only one AI domain for which we found evidence of application to this ST domain.

Additionally, to deepen our discussion, we analyzed the pieces of evidence extracted from the selected secondary studies, to identify more in detail which AI methodology has been applied to specific ST fields. The results of this analysis are reported in Tables 6(a) and 6(b). Each table

Table 6. (b) Second Part of the Resulting Mapping

		Learning															Perception	Integration & Interaction		
		artificial neural network	*boosting*	*classification*	*clustering*	*convolutional neural network*	*decision trees*	*ensemble method*	*probabilistic model*	*recurrent neural network*	*reinforcement learning*	*regression*	*supervised learning*	*support vector machine*	*unsupervised learning*	*image processing*	*intelligent agent*	*q-learning*	*swarm intelligence*
Test Target	*Unit Testing*																		
	Integration Testing										F1								
	System Testing																		
Testing Objective	*Functional Testing*				F12									F12					
	Features Testing																		
	Conformance Testing												F12						
	Non-functional Testing																		
	Usability Testing																		
	Security Testing																		F7
	Safety Testing								F20										
	QoS Testing																		
	Compatibility Testing			F12															
	Reliability Testing								F20										
	Regression Testing			F17	F17				F20		F17	F17							
	Stress Testing																		
	Structural Testing																		
	GUI Testing				F12						F5, F12		F12			F5		F5	
	Acceptance Testing																		
Testing Technique	*Combinatorial Testing*																		
	Requirement-based Testing				F2														
	Mutation Testing			F10, F12	F10, F12							F12			F10, F12				
	Random Testing		F1						F20										
	Concolic Testing																		
	Model-based Testing																		
	Metamorphic-based Testing	F18									F18	F18	F18	F12, F18					
	Equivalence Partitioning						F12												
	Concurrency Testing																		
	Statistical Testing								F20										
Testing Activity	*Test Case Generation*	F5					F5, F12		F5					F5			F4	F5	
	Test Planning				F2				F12										
	Test Oracle Definition	F5, F12, F18	F18	F12		F5	F12	F5			F12	F18	F12, F18	F5		F5			
	Test Case Design and Specification				F12				F12		F12		F12						
	Test Logs Reporting															F5			
	Defect Tracking																		
	Test Case Optimization /Prioritization/Selection	F19	F19	F12, F17, F19	F12, F17, F19		F12, F19		F5, F20	F19	F12, F17, F19	F17, F19	F12, F19	F19	F12, F19				
	Test Results Evaluation	F12			F12			F12	F12			F12			F12				
	Test Data Definition	F18				F5				F5	F5	F18	F18	F18			F4		
	Test Execution			F5	F5						F5								
	Test Repair															F5			
	Flaky Test Prediction								F5										
	Test Environment Development			F12															
	Test Costs Estimation						F12					F12							
Software Testing Fundamentals	*Testing Related Terminology*																		

Rows and columns represent ST and AI concepts, respectively. Cells include the selected secondary studies from which we extracted the evidence of applications.

cell lists the secondary studies in which we found evidence of the application of a specific AI domain/subdomain (column) to support a specific ST domain/field (row). Looking at the mapping at a bird's-eye view, we can observe that: (1) *evolutionary algorithms*, *genetic algorithms*, and *metaheuristic optimisation* have been applied to almost all the ST domains and fields, and (2) *test case generation*, *test oracle definition*, *test case optimization/prioritization/selection*, *test data definition*,

Requirement-based Testing, and *Mutation Testing* are the ST fields that have seen support from most of the AI domains.

Furthermore, to deepen the understanding of RS-RQ3, we drilled down into the cells with several sources assigned to them (i.e., 3 and 4). Such cells indicate that the associated AI concepts have been extensively applied to support the related ST fields. For such cells, in the following bullet list, we summarize (1) the commonalities and differences in the application of the AI techniques identified by the analyzed works, and (2) the difficulties and limitations of the application of the AI techniques to ST objective, and (3) practical insights.

(1) *Evolutionary algorithms* and *genetic algorithms* were found to be the most used AI techniques to support *Mutation testing*, with a prevalence of *genetic algorithms* w.r.t. *evolutionary algorithms*. As an example, F10 [94] reports that an *"...evolutionary strategy with Gaussian Distribution to identify subdomains from which test cases can be selected with higher mutation score"*. While F8 [43] states: *"Genetic Algorithms was also used to address the problem of equivalent mutants in mutation testing."* These two search-based techniques have been applied to *Mutation testing* primarily for two purposes, either for mutant optimization or for *Test case optimization*. However, most of the proposed techniques are either presented in a general manner or are not sufficiently empirically evaluated and can not serve as a base for enabling practitioners to choose a specific technique for a given software. The major challenges include the effort and cost entailed in mutation testing and thus limiting its application to testing real-world programs. As stated by F1 [11] and F16 [57], very few works explored the application of *hyper-heuristics* to *Mutation Testing*, while this technique could bring the advantages of generating stronger mutants and reducing the number of mutants used. As highlighted by F10 [94], meta-heuristic search techniques, and genetic algorithms in particular, have been also effectively applied for the selection of mutant operators and the generation of mutants and generation of test data. From a more practical point of view *Genetic Algorithm*, *Ant Colony*, *Bacteriological Algorithm*, *Hill Climbing*, and *Simulating Annealing* have been extensively used in search-based mutation testing.

(2) *Genetic algorithm* has also been widely used for *Test Case Generation* and *Test Data Definition* as it can be drawn from F9 [89] (*"Researchers apply SBTs for automatic test case generation based on a test objective—adequacy criteria—that is formulated as a fitness function..."*) and F10 [94] (*"The data are considered a pattern to be executed in the design. In the crossover phase, the Genetic Algorithm selects sub-patterns with overlapped inputs to cross and generate new ones..."*). According to F10 [94], experiments conducted in this field showed unsatisfactory results, with the most important challenge being the time necessary for obtaining a good solution, in terms of test cases and test data definition, when more than one solution must be found. However, preliminary results indicate that the use of meta-heuristic search techniques for reducing both the costs and efforts for test data generation in mutation testing is promising. In the *Test Case Generation* field, *genetic algorithms* and *evolutionary algorithms* have been widely applied for "global" search-based techniques (SBTs), i.e., the effective search for global optimal solutions to overcome the problem of getting stuck in local optima. Subsequently, "local" SBTs are used to efficiently obtain the optimal solution starting from global ones. In particular, hill climbing and simulating annealing are the most common examples of local SBTs (F9 [89]). From a more practical point of view, *genetic algorithms* seem to outperform random search in *Test Case Generation* for structural coverage (F11 [3]).

(3) *Metaheuristic optimisation* has been extensively used for *Test Case Optimization/Prioritization/Selection*, *Functional Testing*, *Mutation Testing*, *Test Case Generation*, and *Test Data Definition*. Examples of these applications are reported in F7 [1] (*"Therefore, the initial*

results indicated SA as more effective than other approaches for finding smaller sized test suites"), F9 [89] (*"...surveyed the past work and the current state-of-the-art of the applications of SBTs for structural testing, functional testing..."*), F1 [11] (*"...how SBST has been explored in the context of Mutation Testing, how objective functions are defined and the challenges and opportunities of research in the application of meta-heuristics as search techniques"*), and F8 [43] (*"Test case generation using mutation testing with Ant Colony Optimization," "...a set of test cases are extracted automatically from the textual requirements"*). Although several secondary studies showed that metaheuristic-based techniques have been extensively used to provide solutions for automatizing testing tasks (such as test case selection and test order generation) and for implementing more cost-effective testing processes, some studies, in particular F5 [98] and F11 [3], also highlighted the need for additional empirical experimentation to demonstrate the applicability and the usefulness of metaheuristic in more realistic industrial scenarios.

(4) *Text mining* is the most widely used NLP technique for *Test Case Generation*. As an example, F2 [35] and F5 [98], respectively, report: *"...a set of test cases are extracted automatically from the textual requirements," "...NLP techniques have been used to generate automated test cases from initial requirements documents...a tool named SpecNL generates a description of software test cases in natural language from test case specification...."* As pointed out by F2 [35] and F15 [2], the use of NLP-assisted software testing techniques and tools has been found highly beneficial for researchers and practitioners, as they reduce the cost of test-case generation and the amount of human resources devoted to test activities. However, for a wide industrial usage of NLP-based testing approaches, more work is required to increase their accuracy. Moreover, comparative studies should be performed to highlight strengths and weaknesses of NLP tools and algorithms.

(5) *Ontologies* have been mainly adopted to support the introduction and standardization of terminologies and definitions in ST. Several examples of such application are reported in F3 [96], F4 [25], and F14 [28], respectively: *"...a software testing ontology is designed to represent the necessary software testing knowledge within the software testers' context..."*; *"...the proposed ontology defines a shared vocabulary for testing domain which can be used in knowledge management systems to facilitate communication, integration, search, and representation of test knowledge..."*; *"...the authors presented an ontology, called Test Ontology Model (TOM), to model testing artifacts and relationships between them."* As highlighted by F3 [96], the main benefit of having a suitable software testing ontology is to minimize the heterogeneity, ambiguity and incompleteness problems in terms, properties and relationships. Another potential value of using ontologies and, more in general, semantic web technologies in software testing highlighted by F4 [25] is that they can provide a more powerful mechanism for sharing test assets that are less application-dependent and hence more reusable. By analyzing the terminological coverage of the selected ontologies, in F4 [25] the authors observed that most ontologies cover terms related to dynamic and functional testing. Conversely, only a few ontologies consider terms related to static and non-functional testing. Similarly, the authors of F14 [28] highlighted that most ontologies have limited coverage and none of them is truly a reference ontology or is grounded in a foundational ontology. In conclusion, the software testing community should invest more efforts to get a unique and well-established reference software testing ontology.

(6) *Artificial neural networks* have been used for several testing activities such as oracle definition, test-case generation, test-case refinement, and test-case evaluation. The following evidence of the applications of *artificial neural networks* for *Test Oracle Definition* are reported in F12 [30] and F18 [32], respectively: *"...ML algorithms generated test verdict, metamorphic relation, and—most commonly—expected output oracles. Almost all studies*

employ a supervised or semi-supervised approach, trained on labeled system executions or code metadata–including neural networks, support vector machines, adaptive boosting, and decision trees"; *"...a trend we observed is that the oracle problem tends to be tackled by employing either ANN or decision tree-based approaches...."* Regarding the *Test Oracle Definition* activity, F2 [35] observed that test oracles obtained by using *artificial neural networks* are more efficient, effective, and reusable compared to those generated with existing traditional approaches. Additionally, F12 [30] identified the main advantages of using *artificial neural networks* and *machine learning* in their scalability and in the minimal need of human intervention. As for the main problem faced by researchers when trying to apply *artificial neural networks* and *machine learning* to solve software testing problems, both F12 [30] and F18 [32] identified the need for a substantial amount and high-quality training data, which is the key for *machine learning* algorithms to function as intended.

(7) *Classification*, *clustering*, and *reinforcement learning* AI methodologies have been widely adopted for *Test Case Optimization/Prioritization/Selection*, as highlighted by F19 [74]: *"The main ML techniques used for Test case Selection and Prioritization are: supervised learning (ranking models), unsupervised learning (clustering), reinforcement learning, ...Supervised learning includes all ML techniques that rely on classification or ranking models ...."* Similarly, F17 [50] states that: "...the publication trend of ML technique applied to Test Case Prioritization...shows that the *classification* technique category was the most popular followed by *clustering* then *reinforcement learning* come as the last preferred." F12 [30] also reports *"...approaches that employ reinforcement learning to select and prioritize test cases according to their duration, previous execution and failure history."* F17 [50] reported that *classification* is the most used ML technique as it benefits from the availability of historic data, which results in a high average percentage of faults detected and code coverage effectiveness. F17 [50] also highlighted that *Reinforcement learning* requires a more structured process and improvements before it is mature enough to be included in undergraduate taught programs. Interestingly, F19 [74] highlights that, although *supervised learning*, *unsupervised learning*, *reinforcement learning*, and *natural learning processing* are the four main ML techniques used for test case selection and prioritization, some combinations of them have also been reported in the literature. For example, NLP-based techniques, which are often used for feature preprocessing, were combined with supervised or unsupervised learning to achieve better performance for test case prioritization. F19 [74] highlighted that the lack of standard evaluation procedures and appropriate publicly available datasets resulting from the execution of real world case studies makes it very challenging to draw reliable conclusions concerning ML-based test case selection and prioritization performance. Thus, getting the research community to converge toward common evaluation procedures, metrics, and benchmarks is vital for building a strong body of knowledge we can rely on, without which advancing the state-of-the-art remains an elusive goal.

As a final consideration, we can highlight that the application of *word embedding* to *Test Case Optimization/Prioritization/Selection* has been observed only recently, in 2021 by F19 [74], which reports that *"NLP-based techniques are used for processing textual data, such as topic modeling, Doc2Vec, and LSTM. NLP-based techniques can also be mixed with other ML or non-ML techniques."* Moreover, *word embedding* and neural NLP models are becoming more and more pervasive in transdisciplinary studies and applications, and since *foundation models*[16] are receiving much attention

[16]Rishi Bommasani and Percy Liang (Oct. 2021), Reflections on foundation models: https://thegradient.pub/reflections-on-foundation-models/

Table 7. Future Research Directions Indicated by the Selected Secondary Studies

Future Research Direction	Sources
More rigorous experimental research	F4, F7, F10, F11, F12, F15, F16, F17
Develop evidence with real systems	F12, F16, F18
New data type representation for test data generation	F13
Apply ML to support Automation	F12
Develop an ontology for ST	F14
None	F1, F2, F3, F5, F6, F8, F9, F19, F20

from both academic and industrial researchers, we expect that in the near future NLP will be more extensively applied also to support ST.

5.2.4 RS-RQ4. What are the future research directions of AI in ST? Table 7 summarizes the most recurrent future research directions in AI applied to ST emerging from the analysis of the selected secondary studies, and the list of studies mentioning them. The table was built by analyzing the sentences, extracted from each study, discussing future research directions and grouping sentences indicating similar research directions. Finally, for each group, we defined a category by means of a short summary of the research direction. The need for *more rigorous experimental research* is the most recurrent future research direction (8 of 20 studies). For instance, the authors in F12 [30] state that "most research efforts are not methodologically sound, and some issues remain unexplored." While in F11 [3], the authors report that empirical evidence is needed to assess how "AI-supported techniques (can outperform) current software testing techniques." Three studies identified the need to *develop evidence with real systems*, i.e., to fill the lack of studies investigating the application of AI to ST of larger and more complex software systems. As an example, the authors of F16 [57] observed that "the great majority of the conducted evaluations do not use real and large systems." Similarly, in F12 [30], the authors identified the lack of AI applications to "a wider range of software testing problems." We believe that this future research direction might mitigate the current challenges in the applicability and transferability of AI applications to ST in industrial settings. The authors of F13 [81] identify the need of introducing *new data type representation for test data generation* to apply *genetic algorithms* for automated definition of input/output test values. Another research direction emerging from the analysis is meant to *apply ML to support automation*. The authors from F12 [30] suggest more research be conducted to evaluate how *machine learning* approaches can be used to support ST automation by claiming: "We believe that the overarching motivation for research in this area should be automating most software-testing activities." Moreover, the authors of F14 [28] discuss the necessity to *develop an ontology for ST* as they concluded that "operational versions" of ST taxonomies must be "designed and implemented." Finally, 9 of 20 studies do not propose any future research direction.

6 FURTHER DISCUSSION

In this section, we first provide additional general considerations on the results of our study (Section 6.1). Then, we focus on Testing Activities whose automation has been supported by different AI techniques and, for each AI technique, synthesize the main purpose has been used for (Section 6.2).

6.1 Overall Considerations

Replicability of primary studies: As mentioned in Section 5.2.4, we found that 8 of 20 secondary studies have highlighted the need for rigorous empirical researches to evaluate the outcomes presented by the primary studies. Drawing from this need, we believe that future secondary studies

should devote more attention to this aspect by including specific research questions or quality assessment criteria aimed to evaluate the replicability of the surveyed studies.

Lack of benchmarks about the interplay between AI and ST: We observed the lack of benchmarks that practitioners and researcherscan use to assess the outcomes of applying a specific AI technique to support ST. We feel that this could be an important line of research that can be underpinned by the mapping developed in this study. In particular, benchmarks could include datasets and case studies for which results are already known, and performance metrics the proposed AI-supported ST approaches could be compared against. We also feel that the availability of these benchmarks could facilitate future research advancements by providing a common set of outcomes to outline new research questions and performance metrics.

Use of the mapping from the point of view of ST engineers: ST engineers can use Tables 6(a) and 6(b) to find secondary studies about the AI methodologies that have been already applied to support specific ST domains and concepts. Each non-empty cell indicates that a specific AI concept has been already applied to support a given ST activity or field. For instance, let us suppose we have a practitioner interested in *"Test Data Definition."* The practitioner can look at Tables 6(a) and 6(b) and find out which AI methodologies have been leveraged to support this activity. Moreover, each of the secondary studies reported in non-empty cells supplies pointers to primary studies providing additional details on the specific application of AI in ST. In this specific example, the practitioner interested in the application of *"genetic algorithms"* can deepen this topic by retrieving the primary studies surveyed by the four secondary studies listed in the corresponding cell, i.e., F5, F8, F10, and F11.

Empty cells as food for thought for researchers: Researchers can use the mapping (Tables 6(a) and 6(b)) to identify new research opportunities by inspecting empty cells. An empty cell in these tables means that we did not find evidence of the application of a specific AI concept to a given ST one. Possible *explanations* for empty cells that should be properly taken into account by researchers are:

Explanation 1: There are not enough primary studies on the application of the specific AI concept to a given ST field of interest. As a result, such application has not permeated through the secondary sources and into the resulting mapping of this tertiary study.

Explanation 2: It represents a greenfield opportunity for research, which can be in the form of novel primary studies, or secondary studies that address the mapping associated to the specific empty cell; we note that, similar to this explanation for empty cells, an opportunity to conduct a secondary study is associated to cells of Table 7 including only one study published not recently. As an example, the only secondary study that surveyed the application of AI to support *Non-functional Testing* is F7, that has been published in 2009. As a result, an update of the F7 study could be of interest for the research community.

Explanation 3: It is a false negative for our study. While we have taken great care with the analysis of our secondary sources, there is still the chance that we have missed a reported application.

Explanation 4: It is not possible to apply the specific AI solution to the specific ST problem. The cell might be empty, because the application of AI to software testing might not be feasible. Either temporarily due to limitations in computing power, or by construct, where the application of a specific mapping would not make sense. Researchers must be aware of this possibility when using the mapping as inspiration for research directions.

To exemplify how researchers can use empty cells, let us suppose we are interested to explore the "New Data type representation for test data generation" future research direction reported in Table 7. This future research direction is in relationship with the application of the *knowledge representation reasoning* AI concept to the *Test Data Generation* field; such application corresponds to an empty cell in Table 6 (a). At this point, we can use the rows and columns labels as relevant keywords to perform an initial search in Scopus. To follow this example, we executed this search string ``(TITLE-ABS-KEY (knowledge AND representation AND reasoning) AND TITLE-ABS-KEY (test AND data AND generation))'' in Scopus and it returned *17* studies.[17] We analyzed these papers and derived that just one of them could be potentially related to the empty cell or considered useful for the future research direction we are interested in. As a result, we can argue that this empty cell is consistent to support *Explanation 1*, *Explanation 2*, and *Explanation 3*, and clearly not supportive of *Explanation 4*. If several primary studies related to the empty cell would have been returned from Scopus, then only *Explanation 2* and *Explanation 3* would have been applied.

Use of standard or well-recognized terminologies and taxonomies: We value the use of standard or well-recognized taxonomies (i.e., AI Watch [88] and SWEBOK [18]) as sources of a common language for our domain area. As such, they have been adopted to guide the analysis process. However, our analysis process shows how this outlook is not shared by the community. This puts a toll on the analysis process (in terms of construct validity threats, which we discuss in Section 7) to push the analysis forwards, as an agreement has to be reached upon the term used to describe a phenomenon. Needless to say, we do not view that standards or well-recognized taxonomies need to be static. Not only that these evolve, but novel research proposals might need novel terminology. Yet in general, we observed a lot of variations for concepts that are (or are supposed to be) well understood. We are far from the first to highlight this issue (for instance, see [36, 84]), and in particular at the interplay of AI and software testing, Jöckel et al. [44] highlight how this issue becomes problematic for data analysis in our field and for extending and comparing research results.

6.2 AI Techniques Used to Support the Automation of Testing Activities

As it results from Table 6, several AI techniques have been applied to support ST. In this section, we focus on *Testing Activities* whose automation has been supported by different AI techniques and synthesize the main purpose for which each AI technique has been used for.[18]

AI for Test Case Generation: Secondary studies shared similar conclusions about how AI techniques have been applied to support the test case generation activity. Search-based AI techniques have been used to generate optimal test suites according to a given adequacy criterion, such as code coverage or fault detection. NLP-based techniques have mainly been used to reduce the manual effort of extracting test cases from requirements, specifications and UML models [2, 35, 98]. ML is considered an emerging AI topic for the automation of test case generation. To be applied, these techniques have to learn a behavioral model of the application under test. Such a model is usually built starting from a dataset of inputs and outputs, or on the fly during the exploratory testing of the application under test. The latter approach is mostly used in GUI-based testing, where the user interface is explored and tested at the same time by triggering user events [30]. Ontologies have been used to build a vocabulary of terms that is specific for characterizing the application context of the software under test. The vocabulary can be used to build (1) abstract test cases, i.e.,

[17] June 16, 2022.

[18] Due to space constraints, we limit the discussion to *Testing Activities* that have been supported by more than three AI techniques.

test cases that are not specific to a programming language or (2) platform specific executable test cases, i.e., test cases implemented in a specific programming language. Ontologies have also been used within NLP-assisted test case generation processes to impose restrictions on the context at hand and convert textual documents into an explicit system model for scenario-based test-case generation [25, 35].

AI for Test Case Optimization/Prioritization/Selection: The analyzed studies also pointed out a joint observation that considers the ML-based techniques as the most exploited and promising ones to select or to prioritize the test cases from a test suite for reducing the testing resources such as the testing time and the use of expensive devices [12, 30, 50]. Possible interesting applications of ML show that specific models can be trained, from a dataset of test suites, to select test cases that minimize the testing time or to predict defects in the system under test. The reduction of the testing time allows also the introduction of test case optimization processes based on ML in modern continuous integration development processes. ML leaning-based techniques may also be combined with NLP-based ones. The use of NLP is needed to process textual data for building the dataset used to train the models [74]. Another common conclusion regards the use of ontologies. Semantic web-based techniques are the most used ontologies to define traceability links between test cases, test results and requirements. Such links are exploited to profile the test cases and to select or prioritize the ones that guarantee specific testing adequacy criteria, such as coverage of requirements or failure discovery [25, 96].

AI for Test Data Definition: Most of the secondary studies reached a common consensus that considers the ant colony optimization techniques and genetic algorithms (GAs) as the most cost-effective for the automatic test data definition in the context of mutation testing [3, 43, 81, 89]. GAs have been considered as the most effective solution for the automatic generating of test data for structural, functional, and mutation testing, and it has also been successfully exploited to generate data for testing floating point computations and expert systems. NLP and ML approaches have been mainly used to generate test data for GUI testing and, in particular, for mobile GUI testing. NLP have also been exploited to generate input values expressed in natural language [2, 35], whereas ML techniques (such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Unsupervised and Reinforcement Learning) are used in automated exploratory testing to generate inputs (i.e., user events on the application GUI) allowing the exploration of application states that were not previously visited [32, 98].

AI for Test Oracle Definition: The studies concluded that ML has the potential to solve the "test oracle problem," i.e., the challenge of automatically generating oracles. ML algorithms have been used to generate test verdicts, metamorphic relations, and most commonly expected output oracles [30, 98]. In particular, ML-based predictive models are trained to serve either as a stand-in for an existing test oracle (used to predict a test verdict) or as a way to learn a function that defines expected outputs or metamorphic relationships and that can be used to issue a verdict. Supervised and semi-supervised ML approaches seem to be the most promising; the associated ML models are trained on labeled system executions or on source code metadata. Of these approaches, many use some type of neural networks, such as Backpropagation NNs, Multilayer Perceptrons, RBF NNs, probabilistic NNs, and Deep NNs. Others apply support vector machines, decision trees, and adaptive boosting [32]. The studies showed great promise, but significant open challenges. The performances of the trained ML models are influenced by the quantity, quality, and content of the available training data [32]. Models should be retrained over time. The applied techniques may be insufficient for modeling complex functions with many possible outputs. Research is limited by the overuse of simplistic examples, lack of common benchmarks, and unavailability of code

and data. A robust open benchmark should be created, and researchers should provide replication packages. Computer vision approaches are mainly used to support the oracle definitions in the context of GUI-based testing [98], where the verdicts need that specific regions or images of the graphical user interface are recognized to check their correctness, such as the color, the position on the screen, or the quality of the image.

7 THREATS TO VALIDITY

This section discusses the main possible threats to the validity of our tertiary study, classifying them according to Petersen et al. [77] and drawing suggestions from Zhou et al. [107]. Thus, we classified the Threats to validity into (i) threats to Construct Validity, (ii) threats to internal validity, and (iii) threats to external validity.

Threats to Construct Validity. The use of different terminologies for AI and ST concepts in the selected secondary studies can lead to misclassification. As a strategy to mitigate this possible threat, we started from well-known taxonomies for both the AI [88] and ST [18] domains. In addition, the process of classifying the extracted data was performed iteratively and peer-reviewed by the authors. Furthermore, relevant concepts emerging from secondary studies were added to the adopted reference taxonomies, when missing.

Threats to Internal Validity. One of the major issues with systematic mappings is the risk of missing relevant studies. To mitigate this risk, we adopted a structured process to define and validate our search string, as suggested by Petersen et al. [77], and selected four major digital libraries to execute appropriate queries derived from it. In particular, our search string was designed to retrieve the largest number of published secondary studies by searching for the terms *survey*, *mapping*, *review*, *secondary study*, or *literature analysis* in the title or abstract of the papers. Furthermore, a snowball search process was performed to possibly find additional studies of interest. Another possible threat regards our decision to exclude gray literature papers, such as technical reports and graduate theses, that could lead to miss relevant secondary studies. However, since we reviewed secondary and not primary studies, the risk of excluding relevant but not peer-reviewed material is low. Biases or errors in the application of IC and EC as well as in the quality assessment of papers is another threat to the validity of our study. We mitigated this threat by having each selected paper examined by two groups of co-authors, including an AI expert and an ST specialist each, and having eventual disagreements resolved by face-to-face discussions between the members of the two groups.

Threats to External Validity. Publication bias is another common threat to the validity of secondary and tertiary studies [97]. In particular, the result of our study might have been biased from inaccurate results reported in the selected secondary studies. A common reason for this is that primary studies with negative results are less probable to get accepted for publication and, as a consequence, to be taken into account by secondary studies, and therefore not permeating through to a tertiary study. Another external validity threat for our study relates to the risk of not extracting all the relevant information available in the selected studies or incorrect interpretation of the extracted data. Both these risks may have caused an inaccurate mapping of some analyzed studies. We tried to mitigate this threat by having an AI expert and an ST specialist involved in the data extraction and mapping of each study and resolving eventual disagreements in a face-to-face discussion. Our data extraction could have missed emerging trends provided by recently published primary studies that were not surveyed yet by any secondary studies. Also, since a tertiary study is based on data aggregated in secondary studies, it is possible that relevant information that was

present in primary studies was omitted in the secondary studies and thus missed by our study. This threat is inherent to any tertiary study.

8 CONCLUSIONS

The goal of our tertiary study was to systematically understand how AI has been applied to support ST. As a result, we were able to uncover the interplay between the two domains and to reveal trends and possible future research directions. To achieve this goal, we defined nine RQs (five publication space RQs and four research space RQs) and conducted a systematic mapping study. We designed a strict research protocol and followed a systematic and peer-reviewed process to: (1) select our sources of information, (2) extract evidence from them, and (3) analyze the extracted data to answer our RQs. Starting from an initial set of 877 secondary studies retrieved from four major computer science digital libraries and an additional set of 296 studies retrieved by applying snowballing, the selection process led us to 20 relevant high-quality secondary studies. The analysis of the data extracted from the selected studies let us answer our RQs and derive the following main conclusions.

As for the publication space RQs: (1) the distribution of the selected secondary studies over the publication years (75% of them were published in the past six years), the large amount of unique primary studies they surveyed (710), and the distribution of these primary studies over time (the first dating 1995 and almost two-thirds of them appearing in the past ten years) show a growing interest from the research community in a well-consolidated research topic; (2) most of the selected studies were published in journal venues and a large part of them appeared in top-ranked journals, indicating the high importance of the topic; and (3) most of the authors' affiliations are located in South America (Brazil, Argentina, and Uruguay), while affiliation countries that typically dominate in computer science or computer engineering publications (e.g., USA and China) do not occur in our observations.

Regarding the research space RQs: (1) several AI domains have been applied to support ST with the *Planning* being the most popular one, and *machine learning* and *natural language processing* the most trendy; (2) several ST domains have been supported by AI. Almost all selected secondary studies surveyed the application of AI to the *Testing Activity* ST domain, and a majority of them surveyed the application of AI to the *Testing Technique* domain. Overall, it results that, in recent years, AI has been pervasively introduced in ST; (3) the majority of selected secondary studies investigated the application of *Planning* to support the *Testing Activity*, thus resulting the most surveyed pair of domains; (4) except for *Software Testing Fundamentals*, all ST domains have received support by more than one AI domain; in particular, *Testing Activity* and *Testing Objective* have seen applications from all AI domains. Similarly, by analyzing our mapping at a finer grain level, it results that most ST fields have received support from more than one AI concept, with some concepts having been applied only recently (e.g., *word embedding*); and (5) most frequent future research directions emerging from the selected secondary studies are: (i) the need for more rigorous research, (ii) the evaluation of the proposals in larger or real-world software systems, (iii) more research to evaluate how *machine learning* approaches can be applied to support software testing automation, and (iv) the need for the development of new types of representations to apply genetic algorithms for test data generation.

To the best of our knowledge, this research is the first tertiary study investigating how AI is used to support ST. As a result of this research, we obtained a fine-grained mapping that describes the current interplay between AI and ST. Researchers can leverage this mapping to identify opportunities for future research on new secondary studies to be conducted or new applications of AI to ST to be developed. Practitioners can also use the mapping to take an informed decision on which AI technology to possibly adopt in support of their testing processes.

REFERENCES

[1] Wasif Afzal, Richard Torkar, and Robert Feldt. 2009. A systematic review of search-based testing for non-functional system properties. *Info. Softw. Technol.* 51, 6 (2009), 957–976. https://doi.org/10.1016/j.infsof.2008.12.005

[2] Imran Ahsan, Wasi Haider Butt, Mudassar Adeel Ahmed, and Muhammad Waseem Anwar. 2017. A comprehensive investigation of natural language processing techniques and tools to generate automated test cases. In *Proceedings of the 2nd International Conference on Internet of Things, Data and Cloud Computing (ICC'17)*. ACM, New York, NY, Article 132, 10 pages. https://doi.org/10.1145/3018896.3036375

[3] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, and Rajwinder Kaur Panesar-Walawege. 2010. A systematic review of the application and empirical investigation of search-based test case generation. *IEEE Trans. Softw. Eng.* 36, 6 (2010), 742–762. https://doi.org/10.1109/TSE.2009.52

[4] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Info. Softw. Technol.* 106 (2019), 201–230. https://doi.org/10.1016/j.infsof.2018.10.006

[5] Pardeep Kumar Arora and Rajesh Bhatia. 2018. A systematic review of agent-based test case generation for regression testing. *Arab. J. Sci. Eng.* 43, 2 (Feb. 2018), 447–470. https://doi.org/10.1007/s13369-017-2796-4

[6] Daniel Ashlock. 2006. *Evolutionary Computation for Modeling and Optimization*. Springer Science & Business Media.

[7] J. Bader, S. Seohyun Kim, F. Sifei Luan, S. Chandra, and E. Meijer. 2021. AI in software engineering at Facebook. *IEEE Softw.* 38, 04 (July 2021), 52–61. https://doi.org/10.1109/MS.2021.3061664

[8] Anu Bajaj and Om Sangwan. 2020. *Nature-Inspired Approaches to Test Suite Minimization for Regression Testing*. CRC Press, 99–110. https://doi.org/10.1201/9781003079996-7

[9] Anu Bajaj and Om Prakash Sangwan. 2018. A survey on regression testing using nature-inspired approaches. In *Proceedings of the 4th International Conference on Computing Communication and Automation (ICCCA'18)*. IEEE, 1–5. https://doi.org/10.1109/CCAA.2018.8777692

[10] Anu Bajaj and Om Prakash Sangwan. 2019. A systematic literature review of test case prioritization using genetic algorithms. *IEEE Access* 7 (2019), 126355–126375. https://doi.org/10.1109/ACCESS.2019.2938260

[11] Juliana Marino Balera and Valdivino Alexandre de Santiago Júnior. 2019. A systematic mapping addressing hyper-heuristics within search-based software testing. *Info. Softw. Technol.* 114 (2019), 176–189. https://doi.org/10.1016/j.infsof.2019.06.012

[12] Gerson Barbosa, Érica Ferreira de Souza, Luciana Brasil Rebelo dos Santos, Marlon da Silva, Juliana Marino Balera, and Nandamudi Lankalapalli Vijaykumar. 2022. A systematic literature review on prioritizing software test cases using Markov chains. *Info. Softw. Technol.* 147 (2022), 106902. https://doi.org/10.1016/j.infsof.2022.106902

[13] Feras A. Batarseh, Rasika Mohod, Abhinav Kumar, and Justin Bui. 2020. The application of artificial intelligence in software engineering: A review challenging conventional wisdom. In *Data Democracy*, Feras A. Batarseh and Ruixin Yang (Eds.). Academic Press, 179–232. https://doi.org/10.1016/B978-0-12-818366-3.00010-1

[14] Michael J. Beeson. Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle. 1986. Automated reasoning. Introduction and applications. *J. Symbol. Logic* 51, 2 (1986), 464–465. https://doi.org/10.1017/S0022481200031340

[15] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5, 2 (1994), 157–166. https://doi.org/10.1109/72.279181

[16] Antonia Bertolino. 2007. Software testing research: Achievements, challenges, dreams. In *Proceedings of the Future of Software Engineering (FOSE'07)*. IEEE, 85–103. https://doi.org/10.1109/FOSE.2007.25

[17] Avrim L. Blum and Merrick L. Furst. 1997. Fast planning through planning graph analysis. *Artific. Intell.* 90, 1 (1997), 281–300. https://doi.org/10.1016/S0004-3702(96)00047-1

[18] Pierre Bourque and Richard E. Fairley (Eds.). 2014. *SWEBOK: Guide to the Software Engineering Body of Knowledge* (version 3.0 ed.). IEEE Computer Society, Los Alamitos, CA. Retrieved from http://www.swebok.org/

[19] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. 2013. A survey on optimization metaheuristics. *Info. Sci.* 237 (2013), 82–117. https://doi.org/10.1016/j.ins.2013.02.041

[20] Leo Breiman. 2000. *Bias, Variance, and Arcing Classifiers*. Technical Report 460, Statistics Department, University of California.

[21] Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. 2013. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* 64, 12 (Dec. 2013), 1695–1724. https://doi.org/10.1057/jors.2013.71

[22] Cagatay Catal. 2012. On the application of genetic algorithms for test case prioritization: A systematic literature review. In *Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies (EAST'12)*. ACM, New York, NY, 9–14. https://doi.org/10.1145/2372233.2372238

[23] V. Chandraprakash, J. K. R. Sastry, Gudikandula Sravani, U. S. L. Manasa, A. Khyathi, and A. Harini. 2017. Testing software through genetic algorithms—A survey. *J. Adv. Res. Dynam. Control Syst.* 9 (01 2017), 1607–1633.

[24] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 3 (Sep. 1995), 273–297. https://doi.org/10.1007/BF00994018

[25] Mahboubeh Dadkhah, Saeed Araban, and Samad Paydar. 2020. A systematic literature review on semantic web enabled software testing. *J. Syst. Softw.* 162 (2020), 110485. https://doi.org/10.1016/j.jss.2019.110485

[26] Meenu Dave and Rashmi Agrawal. 2015. Search based techniques and mutation analysis in automatic test case generation: A survey. In *Proceedings of the IEEE International Advance Computing Conference (IACC'15)*. 795–799. https://doi.org/10.1109/IADCC.2015.7154816

[27] Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM* 58, 9 (Aug. 2015), 92–103. https://doi.org/10.1145/2701413

[28] Érica Ferreira de Souza, Ricardo de Almeida Falbo, and Nandamudi L. Vijaykumar. 2013. Ontologies in software testing: A systematic literature review. In *Proceedings of the 6th Seminar on Ontology Research in Brazil (CEUR Workshop Proceedings, Vol. 1041)*, Marcello Peixoto Bax, Mauricio Barcellos Almeida, and Renata Wassermann (Eds.). CEUR-WS.org, 71–82. http://ceur-ws.org/Vol-1041/ontobras-2013_paper25.pdf

[29] Dursun Delen and Martin D. Crossland. 2008. Seeding the survey and analysis of research literature with text mining. *Expert Syst. Appl.* 34, 3 (2008), 1707–1720. https://doi.org/10.1016/j.eswa.2007.01.035

[30] Vinicius H. S. Durelli, Rafael S. Durelli, Simone S. Borges, Andre T. Endo, Marcelo M. Eler, Diego R. C. Dias, and Marcelo P. Guimarães. 2019. Machine learning applied to software testing: A systematic mapping study. *IEEE Trans. Reliabil.* 68, 3 (2019), 1189–1212. https://doi.org/10.1109/TR.2019.2892517

[31] Dieter Fensel. 2001. *Ontologies*. Springer, Berlin, 11–18. https://doi.org/10.1007/978-3-662-04396-7_2

[32] Afonso Fontes and Gregory Gay. 2021. Using machine learning to generate test oracles: A systematic literature review. In *Proceedings of the 1st International Workshop on Test Oracles (TORACLE'21)*. ACM, New York, NY, 1–10. https://doi.org/10.1145/3472675.3473974

[33] International Organization for Standardization. 2010. ISO/IEC/IEEE international standard-systems and software engineering-vocabulary. *ISO/IEC/IEEE 24765:2010(E)* (2010), 418. https://doi.org/10.1109/IEEESTD.2010.5733835

[34] International Organization for Standardization. 2013. ISO/IEC/IEEE international standard-software and systems engineering–software testing–Part 1: Concepts and definitions. *ISO/IEC/IEEE 29119-1:2013(E)* (2013), 64. https://doi.org/10.1109/IEEESTD.2013.6588537

[35] Vahid Garousi, Sara Bauer, and Michael Felderer. 2020. NLP-assisted software testing: A systematic mapping of the literature. *Info. Softw. Technol.* 126 (2020), 106321. https://doi.org/10.1016/j.infsof.2020.106321

[36] E. Georgiadou. 2018. Reflections on the need for disambiguation of terminology for software process improvement. In *Systems, Software and Services Process Improvement. EuroSPI 2018. Communications in Computer and Information Science*, X. Larrucea, I. Santamaria, R. O'Connor, and R. Messnarz (Eds.). Vol. 896, Springer, Cham, 577–589. https://doi.org/10.1007/978-3-319-97925-0_49

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org

[38] Michael Haenlein and Andreas Kaplan. 2019. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *Calif. Manage. Rev.* 61, 4 (2019), 5–14. https://doi.org/10.1177/0008125619864925

[39] Daniel A. Hashimoto, Guy Rosman, Daniela Rus, and Ozanan R. Meireles. 2018. Artificial intelligence in surgery: Promises and perils. *Ann. Surg.* 268, 1 (2018), 70. https://doi.org/10.1097/SLA.0000000000002693

[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer, New York. https://doi.org/10.1007/978-0-387-84858-7

[41] J. J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.* 79, 8 (Apr. 1982), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

[42] Junyan Hu, Ali E. Turgut, Tomáš Krajník, Barry Lennox, and Farshad Arvin, 2022. Occlusion-based coordination protocol design for autonomous robotic shepherding tasks. *IEEE Transactions on Cognitive and Developmental Systems* 14, 1 (March 2022), 126–135. https://doi.org/10.1109/TCDS.2020.3018549

[43] Nishtha Jatana, Bharti Suri, and Shweta Rani. 2017. Systematic literature review on search based mutation testing. *e-Inform. Softw. Eng. J.* 11, 1 (2017), 59–76. https://doi.org/10.5277/e-Inf170103

[44] Lisa Jöckel, Thomas Bauer, Michael Kläs, Marc P. Hauer, and Janek Groß. 2021. Towards a common testing terminology for software engineering and data science experts. In *Proceedings of the Product-Focused Software Process Improvement: 22nd International Conference (PROFES'21)*, Turin, Italy, Springer-Verlag, Berlin, Heidelberg, 281–289. https://doi.org/10.1007/978-3-030-91452-3_19

[45] Dan Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, Upper Saddle River, NJ. Retrieved from http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y

[46] L. P. Kaelbling, M. L. Littman, and A. W. Moore. 1996. Reinforcement learning: A survey. *J. Artific. Intell. Res.* 4 (1996), 237–285. https://doi.org/10.1613/jair.301

[47] Meir Kalech, Rui Abreu, and Mark Last. 2021. *Artificial Intelligence Methods for Software Engineering*. World Scientific, Singapore. https://doi.org/10.1142/12360

[48] Manju Khari and Prabhat Kumar. 2019. An extensive evaluation of search-based software testing: A review. *Soft Comput.* 23, 6 (Mar. 2019), 1933–1946. https://doi.org/10.1007/s00500-017-2906-y

[49] M. Khari and Kumar M. Vaishali. 2019. Search-based secure software testing: A survey. In *Software Engineering. Advances in Intelligent Systems and Computing*, M. Hoda, N. Chauhan, S. Quadri, and P. Srivastava (Eds.). Vol. 731, Springer, Singapore. 375–381. https://doi.org/10.1007/978-981-10-8848-3_35

[50] Muhammad Khatibsyarbini, Mohd Adham Isa, Dayang N. A. Jawawi, Muhammad Luqman Mohd Shafie, Wan Mohd Nasir Wan-Kadir, Haza Nuzly Abdull Hamed, and Muhammad Dhiauddin Mohamed Suffian. 2021. Trend application of machine learning in test case prioritization: A review on techniques. *IEEE Access* 9 (2021), 166262–166282. https://doi.org/10.1109/ACCESS.2021.3135508

[51] Tariq M. King, Jason Arbon, Dionny Santiago, David Adamo, Wendy Chin, and Ram Shanmugam. 2019. AI for testing today and tomorrow: Industry perspectives. In *Proceedings of the IEEE International Conference On Artificial Intelligence Testing (AITest'19)*. IEEE, 81–88. https://doi.org/10.1109/AITest.2019.000-3

[52] B. Kitchenham and S. Charters. 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

[53] Barbara Kitchenham, Rialette Pretorius, David Budgen, O. Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. 2010. Systematic literature reviews in software engineering—A tertiary study. *Info. Softw. Technol.* 52, 8 (2010), 792–805. https://doi.org/10.1016/j.infsof.2010.03.006

[54] Vegard Kolbjørnsrud, Richard Amico, and Robert J. Thomas. 2016. How artificial intelligence will redefine management. *Harvard Bus. Rev.* 2 (2016), 1–6.

[55] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. 2006. Machine learning: A review of classification and combining techniques. *Artific. Intell. Rev.* 26, 3 (Nov. 2006), 159–190. https://doi.org/10.1007/s10462-007-9052-3

[56] Gerhard Lakemeyer and Bernhard Nebel. 1994. *Foundations of Knowledge Representation and Reasoning*. Springer, Berlin, 1–12. https://doi.org/10.1007/3-540-58107-3_1

[57] Jackson A. Prado Lima and Silvia Regina Vergilio. 2018. Search-based higher order mutation testing: A mapping study. In *Proceedings of the 3rd Brazilian Symposium on Systematic and Automated Software Testing (SAST'18)*. ACM, New York, NY, 87–96. https://doi.org/10.1145/3266003.3266013

[58] Rui Lima, António Miguel Rosado da Cruz, and Jorge Ribeiro. 2020. Artificial intelligence applied to software testing: A literature review. In *Proceedings of the 15th Iberian Conference on Information Systems and Technologies (CISTI'20)*. 1–6. https://doi.org/10.23919/CISTI49556.2020.9141124

[59] M. Craglia, A. Annoni, P. Benczur, P. Bertoldi, B. Delipetrev, G. De Prato, C. Feijoo, E. Fernandez Macias, E. Gomez Gutierrez, M. Iglesias Portela, H. Junklewitz, M. Lopez Cobo, B. Martens, S. Figueiredo Do Nascimento, S. Nativi, A. Polvora, J. I. Sanchez Martin, S. Tolan, I. Tuomi, and L. Vesnic Alujevic. 2018. *Artificial Intelligence: A European Perspective*. Technical Report KJ-NA-29425-EN-N. Luxembourg. https://doi.org/10.2760/11251

[60] Ruchika Malhotra and Manju Khari. 2013. Heuristic search-based approach for automated test data generation: A survey. *Int. J. Bio-Inspired Comput.* 5, 1 (Apr. 2013), 1–18. https://doi.org/10.1504/IJBIC.2013.053045

[61] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

[62] Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA. Retrieved from http://nlp.stanford.edu/fsnlp/.

[63] Timo Mantere and Jarmo T. Alander. 2005. Evolutionary software engineering, a review. *Appl. Soft Comput.* 5, 3 (2005), 315–331. https://doi.org/10.1016/j.asoc.2004.08.004

[64] Santiago Matalonga, Domenico Amalfitano, Andrea Doreste, Anna Rita Fasolino, and Guilherme Horta Travassos. 2022. Alternatives for testing of context-aware software systems in non-academic settings: Results from a rapid review. *Info. Softw. Technol.* 149 (2022), 106937. https://doi.org/10.1016/j.infsof.2022.106937

[65] Phil McMinn. 2004. Search-based software test data generation: A survey. *Softw. Test., Verific. Reliabil.* 14, 2 (2004), 105–156. https://doi.org/10.1002/stvr.294

[66] Carlo Milana and Arvind Ashta. 2021. Artificial intelligence techniques in finance and financial markets: A survey of the literature. *Strategic Change* 30 (05 2021), 189–209. https://doi.org/10.1002/jsc.2403

[67] Deepti Bala Mishra, Arup Abhinna Acharya, and Rajashree Mishra. 2019. Evolutionary algorithms for path coverage test data generation and optimization: A review. *Indonesian J. Electr. Eng. Comput. Sci.* 15, 1 (July 2019), 504. https://doi.org/10.11591/ijeecs.v15.i1.pp504-510

[68] D. B. Mishra, R. Mishra, K. N. Das, and A. A. Acharya. 2017. A systematic review of software testing using evolutionary techniques. In *Proceedings of 6th International Conference on Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing*, K. Deep et al. (Eds.). Vol. 546, Springer, Singapore, 174–184. https://doi.org/10.1007/978-981-10-3322-3_16

[69] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT Press. https://doi.org/10.7551/mitpress/3927.001.0001

[70] Bernard M. E. Moret. 1982. Decision trees and diagrams. *ACM Comput. Surv.* 14, 4 (Dec. 1982), 593–623. https://doi.org/10.1145/356893.356898

[71] Ana Filipa Nogueira, José Ribeiro, Francisco Vega, and Mário Zenha-Rela. 2014. Object-oriented evolutionary testing: A review of evolutionary approaches to the generation of test data for object-oriented software. *Int. J. Nat. Comput. Res.* 4 (Oct. 2014), 15–35. https://doi.org/10.4018/ijncr.2014100102

[72] Vilém Novák, Irina Perfilieva, and Jiří Močkoř. 1999. *Mathematical Principles of Fuzzy Logic*. Springer. https://doi.org/10.1007/978-1-4615-5217-8

[73] D. Opitz and R. Maclin. 1999. Popular ensemble methods: An empirical study. *J. Artific. Intell. Res.* 11 (Aug. 1999), 169–198. https://doi.org/10.1613/jair.614

[74] Rongqi Pan, Mojtaba Bagherzadeh, Taher A. Ghaleb, and Lionel Briand. 2021. Test case selection and prioritization using machine learning: A systematic literature review. *Empir. Softw. Eng.* 27, 2 (Dec. 2021), 29. https://doi.org/10.1007/s10664-021-10066-6

[75] Thomas B. Passin. 2004. *Explorer's Guide to the Semantic Web*. Manning Publications.

[76] P. Paygude and S. D. Joshi. 2020. Use of evolutionary algorithm in regression test case prioritization: A review. In *Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI'18). Lecture Notes on Data Engineering and Communications Technologies*, A. Pandian, T. Senjyu, S. Islam, and H. Wang (Eds.). Vol. 31, Springer, Cham, 56–66. https://doi.org/10.1007/978-3-030-24643-3_6

[77] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Info. Softw. Technol.* 64 (2015), 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[78] Ioannis Pitas. 2000. *Digital Image Processing Algorithms and Applications*. John Wiley & Sons.

[79] V. Chandra Prakash, Subhash Tatale, Vrushali Kondhalkar, and Laxmi Bewoor. 2018. A critical review on automated test case generation for conducting combinatorial testing using particle swarm optimization. *Int. J. Eng. Technol.* 7, 3.8 (2018), 22–28. https://doi.org/10.14419/ijet.v7i3.8.15212

[80] Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press. https://doi.org/10.1017/CBO9780511519857

[81] Davi Silva Rodrigues, Márcio Eduardo Delamaro, Cléber Gimenez Corrêa, and Fátima L. S. Nunes. 2018. Using genetic algorithms in test data generation: A critical systematic mapping. *ACM Comput. Surv.* 51, 2, Article 41 (May 2018), 23 pages. https://doi.org/10.1145/3182659

[82] Daniel Rodriguez, Javier Dolado, and Javier Tuya. 2015. Bayesian concepts in software testing: An initial review. In *Proceedings of the 6th International Workshop on Automating Test Case Design, Selection and Evaluation (A-TEST'15)*. ACM, New York, NY, 41–46. https://doi.org/10.1145/2804322.2804329

[83] Rakesh Roshan, Dr. Rabins Porwal, and Chandramani Sharma. 2012. Review of search based techniques in software testing. *Int. J. Comput. Appl.* 51 (08 2012), 42–45. https://doi.org/10.5120/8050-1387

[84] Terence P. Rout. 1999. Consistency and conflict in terminology in software engineering standards. In *Proceedings of the 4th IEEE International Software Engineering Standards Symposium and Forum (ISESS'99)*. IEEE, 67–74. https://doi.org/10.1109/SESS.1999.766579

[85] Winston W. Royce. 1987. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering (ICSE'87)*. IEEE Computer Society Press, Washington, DC, 328–338.

[86] David E. Rumelhart, Bernard Widrow, and Michael A. Lehr. 1994. The basic ideas in neural networks. *Commun. ACM* 37, 3 (Mar. 1994), 87–92. https://doi.org/10.1145/175247.175256

[87] Stuart Russell and Peter Norvig. 2016. *Artificial Intelligence: A Modern Approach*. Pearson. Retrieved from https://books.google.it/books?id=XS9CjwEACAAJ

[88] S. Samoili, M. Lopez Cobo, E. Gomez Gutierrez, G. De Prato, F. Martinez-Plumed, and B. Delipetrev. 2020. Defining artificial intelligence: Towards an operational definition and taxonomy of artificial intelligence. Retrieved from https://op.europa.eu/en/publication-detail/-/publication/6cc0f1b6-59dd-11ea-8b81-01aa75ed71a1/language-en. https://doi.org/10.2760/382730

[89] Aneesa Saeed, Siti Hafizah Ab Hamid, and Mumtaz Begum Mustafa. 2016. The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review. *Appl. Soft Comput.* 49 (2016), 1094–1117. https://doi.org/10.1016/j.asoc.2016.08.030

[90] Sunita Sarawagi. 2008. Information extraction. *Found. Trends Databases* 1, 3 (2008), 261–377. https://doi.org/10.1561/1900000003

[91] M. Edgar Serna, M. Eder Acevedo, and Alexei Serna A. 2019. Integration of properties of virtual reality, artificial neural networks, and artificial intelligence in the automation of software tests: A review. *J. Softw. Evol. Process* 31, 7 (July 2019), 12 pages. https://doi.org/10.1002/smr.2159

[92] Manoj Sharma, Arun Sharma, and R. Kumar. 2011. Soft computing-based software test cases optimization: A survey. *Int. Rev. Comput. Softw.* 6 (07 2011), 512–526.
[93] Riya Sil, Bharat Bhushan, and Arun Majumdar. 2019. Artificial intelligence and machine learning based legal application: The state-of-the-art and future research trends. In *Proceedings of the International Conference on Computing, Communication, and Intelligent Systems (ICCCIS'19)*. IEEE, 57–62. https://doi.org/10.1109/ICCCIS48478.2019.8974479
[94] Rodolfo Adamshuk Silva, Simone do Rocio Senger de Souza, and Paulo Sérgio Lopes de Souza. 2017. A systematic review on search based mutation testing. *Info. Softw. Technol.* 81 (2017), 19–35. https://doi.org/10.1016/j.infsof.2016.01.017
[95] Bharti Suri and Shweta Singhal. 2012. Literature survey of Ant colony optimization in software testing. In *Proceedings of the CSI 6th International Conference on Software Engineering (CONSEG'12)*. IEEE, 1–7. https://doi.org/10.1109/CONSEG.2012.6349501
[96] Guido Tebes, Denis Peppino, Pablo Becker, Gerardo Matturro, Martin Solari, and Luis Olsina. 2020. Analyzing and documenting the systematic review results of software testing ontologies. *Info. Softw. Technol.* 123 (2020), 106298. https://doi.org/10.1016/j.infsof.2020.106298
[97] Huynh Khanh Vi Tran, Michael Unterkalmsteiner, Jürgen Börstler, and Nauman bin Ali. 2021. Assessing test artifact quality—A tertiary study. *Info. Softw. Technol.* 139 (2021), 106620. https://doi.org/10.1016/j.infsof.2021.106620
[98] Anna Trudova, Michal Dolezel, and Alena Buchalcevova. 2020. Artificial intelligence in software test automation: A systematic literature review. In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'20)*. INSTICC, SciTePress, 181–192. https://doi.org/10.5220/0009417801810192
[99] Edward Tsang. 1993. *Foundations of Constraint Satisfaction*. Elsevier. https://doi.org/10.1016/c2013-0-07627-x
[100] Derya Yeliz Ulutaş. 2020. The generation of optimized test data: Preliminary analysis of a systematic mapping study. In *Proceedings of the Turkish National Software Engineering Symposium (UYMS'20)*. IEEE, 1–4. https://doi.org/10.1109/UYMS50627.2020.9247014
[101] Vandana and Ajmer Singh. 2017. Multi-objective test case minimization using evolutionary algorithms: A review. In *Proceedings of the International Conference of Electronics, Communication and Aerospace Technology (ICECA'17)*, Vol. 1. IEEE, 329–334. https://doi.org/10.1109/ICECA.2017.8203698
[102] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Mach. Learn.* 8, 3 (01 May 1992), 279–292. https://doi.org/10.1007/BF00992698
[103] George M. White. 1995. Natural language understanding and speech recognition. In *Readings in Human–Computer Interaction*, Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, and Saul Greenberg (Eds.). Morgan Kaufmann, 554–563. https://doi.org/10.1016/B978-0-08-051574-8.50058-3
[104] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE'14)*. ACM, New York, NY, Article 38, 10 pages. https://doi.org/10.1145/2601248.2601268
[105] Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. *Ann. Data Sci.* 2, 2 (2015), 165–193. https://doi.org/10.1007/s40745-015-0040-1
[106] G. Udny Yule. 1897. On the theory of correlation. *J. Roy. Stat. Soc.* 60, 4 (1897), 812–854. https://doi.org/10.1111/j.2397-2335.1897.tb02784.x
[107] Xin Zhou, Yuqin Jin, He Zhang, Shanshan Li, and Xin Huang. 2016. A map of threats to validity of systematic literature reviews in software engineering. In *Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC'16)*. IEEE, 153–160. https://doi.org/10.1109/APSEC.2016.031

Received 19 July 2022; revised 25 May 2023; accepted 3 August 2023