Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

JOURNAL SUBMISSION FOR JSEP SPECIAL ISSUE

A Survey of the State-of-the-Art Approaches for Evaluating Trust in Software Ecosystems

Fang, Hou[1] | **Slinger, Jansen**[1,2]

[1]Department of Information and Computer Science, Utrecht University, Utrecht, Netherlands

[2]School of Engineering Science, Lappeenranta University of Technology, Lappeenranta, Finland

Correspondence
Fang, Hou
Email: f.hou@uu.nl

Funding Information
The authors have no relevant financial interest in this article.

Abstract

Third-party software has streamlined the software engineering process, allowed software engineers to focus on developing more advanced components, and reduced time and cost. This shift has led to software development strategies moving from competition to collaboration, resulting in the concept of software ecosystems, in which internal and external actors work together on shared platforms and place their trust in the ecosystem. However, increased shared components have also created challenges, especially in security, as the large dependency trees significantly enlarge a system's attack surface. The situation is made worse by the lack of effective ways to measure and ensure the trustworthiness of these components.

In this article, we explore current approaches used to evaluate trust in software ecosystems, focusing on analyzing the specific techniques utilized, the primary factors in trust evaluation, the diverse formats for result presentation, as well as the software ecosystem entities considered in the approaches. Our goal is to provide the status of current trust evaluation approaches, including their limitations. We identify key challenges, including the limited coverage of software ecosystem entities, the objectivity, universality, and environmental impacts of the evaluation approaches, the risk assessment for the evaluation approaches, and the security attacks posed by trust evaluation in these approaches.

KEYWORDS
Software Ecosystem, Software Trust, Trust Evaluation

FINANCIAL DISCLOSURE
The authors have no relevant financial interest in this article.

CONFLICT OF INTEREST
The authors declare no potential conflict of interest.

1 | INTRODUCTION

To maximize efficiency and profitability, every software engineer builds software products using other software components or tools throughout the entire software lifecycle. Software market conditions encourage competition, collaboration, and sharing. By sharing an underlying technical platform, actors who collaborate to serve a market for software and services form software ecosystems (SECOs)[43]. SECOs enable software solutions to be created by composing software components developed by both internal and external actors[44]. A worldwide SECO is a network of interconnected software applications, platforms, tools, and services that together form a dynamic and complex environment. It provides a platform for global software stakeholders to communicate. However, this communication attracts actors working collaterally to create software solutions, and more attackers are committed to spreading more vulnerabilities through the software supply chain. According to Sonatype (2022), over the past three years, malicious software supply chain attacks have increased by a further 633%, averaging a 742% annual increase[45]. *Trust in SECOs is being gradually eroded.*

 © 2023 Copyright Holder Name

Despite the wide variety of user requirements, unprincipled trust seems to be placed by the software engineers, especially open-source software (OSS) engineers in the products provided by software-producing organizations, as long as they promise they can fulfill their requirements[46]. Our previous work conducted interviews with 24 experts in OSS and closed software selection with five to 26 years of working experience in their respective industries[47]. The result reveals that OSS selection rarely employs formal selection protocols, but closed software follows formal protocols. For OSS interviewees, the projects they supported do not usually involve a high level of risk or importance. Usually, they refer to some information, standards, or facts to assess trust in their software components selection process, such as GitHub stars, the number of contributors, the ratio of open to closed issues, compatibility, program language, and user reviews of the software-producing organization, with little regard for the software trust, such as whether the software has been well tested, will protect user privacy, maintain confidentiality, or mitigate vulnerabilities. This is because they believe that SECO hubs, such as the Package Manager, or APP, provide services to ensure that the software components they download are trustworthy. For closed software selection, the selection factors are different from the OSS selection factors. The selection focuses on business rather than technology. Software-producing organizations are also a major focus, with attention to detailed aspects such as brand, reputation, and certification. Moreover, when selecting closed software, there is a preference for mature, stable, and low-risk products over cutting-edge technology. Other essential considerations include the product category, how well the software suits its intended purpose, and the availability of maintenance and support following implementation.

Selecting software equates to selecting a SECO[47]. According to our previous systematic literature review (SLR), SECO trust is composed of all upstream and downstream trust between its actors and hubs[48]. It forms the foundation of this study, indicating that software selection or evaluation should not be limited to software quality or services per se, but should be extended to hubs and actors throughout the SECO. This study extends the SECO trust topic, to comprehend the research challenges faced in trust evaluation within SECO, specifically in software or SECO selection and evaluation. To achieve this, we conducted systematic mappings (SM) to examine the approaches adopted in evaluating trust in SECO and check whether these approaches adequately consider the trust relationships between the hubs and actors across the upstream and downstream. Through a detailed examination of these state-of-the-art approaches, we attempt to uncover how these approaches function. On this basis, we analyze the outstanding challenges and future direction in the trust evaluation design. Figure 1 presents a brief overview of this study.

The rest of this paper is structured as follows. Section 2 explains the concept of trust, sources of trust facts, trust attributes, common trust evaluation approaches, and practice in software evaluation. Section 3 introduces the research method. We conducted systematic mappings as an acquisition process to collect the SECO trust evaluation approaches. Section 4 reports the state-of-the-art SECO trust approaches in terms of evaluation techniques, evaluation criteria, presentation of evaluation results, and covered SECO entities. In section 5, we analyze the outstanding challenges and our future plan. Finally, in Section 6, we conclude that current research on trust in SECO focuses exclusively on the software. Future research should be extended to include more hubs and actors within or across SECOs.

2 | PREREQUISITES

2.1 | Notion of Trust

Trust, often defined as confidence, expectation, willingness, or belief, encompasses both emotional and cognitive dimensions[49]. Emotional aspects of trust are often subjective and influenced by personal experiences and perceptions. Predictions in the cognitive dimension are based on empirical and historical data[50]. Consequently, measuring trust is often challenging, as it is not universally perceived or demanded by every individual or organization. It is possible to interpret trust in terms of some similar characteristics. For instance, *reputation*, complements trust and can be seen as public perception of trustworthiness[51]. *Credibility* is another characteristic, describing the trustworthiness of information[52]. *Dependability* is inherent, as trust enables us to depend on others. At its core, trust involves anticipating risk, uncertainty, and vulnerability. Due to uncertainty and vulnerability, trust becomes all the more important. Additionally, trust evolves over time[53].

Based on the literature review on trust in SECO, SECO trust is a broader concept, including, for example, the trust of end-users in the software engineers to build solutions to their (business) problems, and the trust of software engineers in the package management system to deliver their software correctly to end-users. In this article, we refer to these stakeholders with a need for trust as **trust information consumers**. The study of trust in SECO should start with trust in SECO entities, such as software components, and extend to trust in the marketplace and society, as well as trust in the global ecosystem[54], for instance, brand trust, and product category trust. Software trust refers to the willingness of actors to accept risks based on subjective beliefs[55]. It

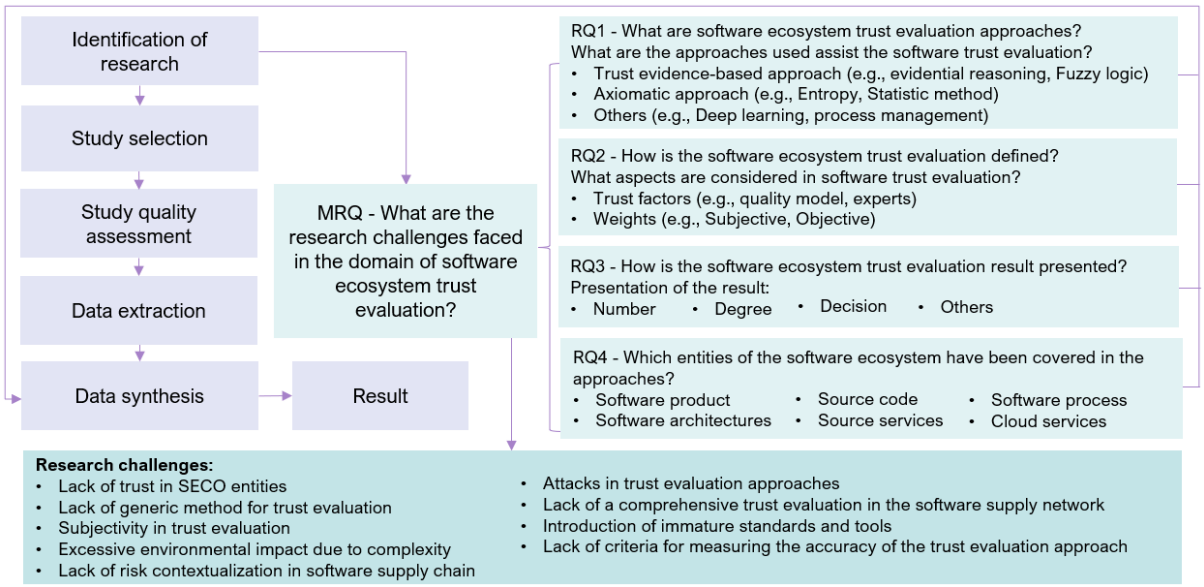

FIGURE 1 *This figure provides an overview of this study, including the research method steps and the findings*

represents an upstream trust in which actors expect assurance that others within a common technology platform will exhibit reliable behavior and provide valuable software products[48]. There are both emotional and cognitive dimensions to software trust. It is often generated based on the observation of a set of historical data and individuals' cognition of software products[56]. In line with ISO/IEC 15408, a software entity is deemed trustworthy if its components, operations, and processes remain predictable under all conditions and can withstand interference from viruses and external factors[36]. According to the interviews with closed software selection experts, selecting a software product is the equivalent of selecting a SECO[47]. Consequently, software trust factors are often the same as SECO trust factors. This is the basis of this study.

2.2 | Trust Fact Sources

Trust is a result of certain facts and beliefs. Trust fact sources determine the technology used by the trust evaluation approaches. It is possible to increase the reliability of the calculated trust by adopting smart information sources, but this will result in much complexity of the approach[57].

There are several types of trust fact sources. (1) Experience, including direct experience and indirect experience. Direct experience includes direct interaction with the software products. Indirect experience is based on an observation of the interaction between other end-users and software products[48]. Usually, recommender systems adopt this type of information source. (2) Trust evidence, which includes the information that can be collected from software-producing organizations and engineers, third-party software components communities, data mining tools, or some questions and answers portals, e.g., StackOverflow. Computational trust and reputation models usually rely on both experience and evidence information sources[57]. Li et al. (2022) provide more specific sources of information based on a set of interviewees, including version control systems, such as GitHub and SourceForge; issue tracking systems, such as GitHub Issues and Jira; question and answer portals, such as StackOverflow and Reddit; forum and blogs, such as Medium and Hackernews; and security platform, such as CVE and NVD[58].

2.3 | SECO Trust Attributes

TCPA was launched in 1999 by IBM, HP, Intel, and Microsoft, and renamed Trusted Computing Group (TCG) in 2003, resulting in a new upsurge of trust in computing. TCG has been addressing trust issues, particularly in the context of PCs, servers, networking gear, and embedded systems, driven by the Trusted Platform Module (TPM) specification[59]. It is important to note that TCG does not only consider information secrecy, but also emphasizes the integrity and authenticity of the information, and makes trusted computing more industrialized and comprehensive[60].

Several organizations are also working to introduce specifications on software trust. For instance, ISO/IEC 9126:2001 and ISO/IEC 25010. The attributes are based on a broad spectrum, including quality, functionality, performance, dependability, reliability, reputation, security, or credibility. Quality of Service (QoS) is another common attribute, the higher the quality, the higher the trustworthiness [11]. Table 1 shows the major attributes of software trust based on our studies [48,47]. Those attributes stem from the interview results with 24 software selection experts and an SM from 112 selected peer-reviewed articles. Results show that software providers, quality of the source code, security, reliability of the software product, documentation, and structural guarantees (e.g., licenses, and contracts) are the most important factors.

Security and vulnerability should be emphasized. Because the essence of trust is the anticipation of risk, uncertainty, and vulnerability. Hence, trustworthy software products are those that maximize effectiveness or minimize risk [50] and uncertainty [61], ensuring no exploitable vulnerabilities exist either of malicious or unintended origin [62] under uncertain and unreliable environment conditions [6]. Trusted Computer National Evaluation Criteria (TCSEC) restricted trustworthiness to security as the only attribute to consider [38]. SECO trust can often be attributed to various quality attributes rather than a single factor [63].

TABLE 1 The summary of software trustworthiness attributes. Rankings are based on the number of mentions (top to bottom).

Open-Source Software	Closed Software	Literature Review
Documentation	Software Provider Organization	Source Code
Contributor	Data Integrity	Reliability
Maintenance/Support	Software Testing	Security
Code Quality	Certification	Documentation
Software Provider Organization	Functionality	License
Dependency	Maintenance/Support	Maintainability
Compatibility	Language/Framework	Process
Integration	Sensitive Information	Usability
GitHub Issues	Vulnerability	Vulnerabilities
Testing	Cost	Popularity
Vulnerability	Code Quality	Framework
Community	License	Performance

Red-Technical Factors; Gray-Organizational Factors; Blue-Structural Assurance Factors

2.4 | Common Trust Evaluation Approaches

Trust is based on subjective judgments. Subjective judgments, in turn, are derived from the experience of the users, and often experience is taken for granted as being correct. There are the following prerequisites for the evaluation: (1) trust evaluation is based on a set of observations, including experience and evidence; (2) different experience and evidence has different effects on trust; (3) trust has scales, which means that trust is comparable; and (4) the more reliable the available evidence, the more reliable the trust score [50].

There are two types of trust derived from user experience: direct trust (based on the user's experience) and recommendation trust (based on the experiences of others) [64]. For direct trust, trust built up over time as a result of repeated interactions between the trustor and trustee. Reputation can be defined as a measure of the trustworthiness of an individual based on recommendations or ratings from other members of the community [65]. Reputation system often refers to the management of trust in a large social network, among which a participant's trust is determined by the feedback they receive from their peers [66]. However, the issue is the need to confirm the validity of the recommendation [50]. Centralized reputation systems collect all ratings, generate reputation scores for each participant, and make all scores publicly available [65]. For instance, eBay, where customers and sellers rate each other using numerical ratings or feedback comments, and it emphasizes reputation in auctions, based on theory and evidence from eBay, as well as rating negatively dishonest behavior [67,68]. Distributed reputation systems allow participants to obtain ratings from other members of the community. Each individual agent derives a reputation score for the target party based on the ratings received and other possible information [65]. Usually distributed reputation systems require a social network to support trust in "virtual organizations of shared resources" [69,70]. For instance, ReGreT [71], is a modular trust and reputation system oriented to complex small or middle size e-commerce environments where social relations among individuals play an important role. By assigning different weights to trust facts and factors, the axiomatic approach assesses trust from a mathematical perspective.

Recommendation techniques [72] mainly include collaborative filtering (CF), content-based (CB), knowledge-based (KB), and computational intelligence (CI) techniques. CF-based recommendation techniques, work by collecting user feedback in the form of ratings for items in a given domain and exploiting the most similar profiles of several users among others in determining how to recommend an item [72,73,74]. The item-based CF approach [75] takes into account the items the target user has rated and determines how similar they are to the target item. CB techniques mainly depend on the content and relevant profile descriptions to generate personalized recommendations [72]. KB techniques provide users with advice on making decisions or taking action. For hybrid-based techniques, multiple basic mechanisms of combinations are used in recommender systems, including weighting, mixing, switching, feature combination, feature augmentation, cascade, and meta-level. CI techniques include Bayesian techniques, artificial neural networks, clustering techniques, genetic algorithms, and fuzzy set techniques.

Trust is used to translate choices into decisions [50]. That is the reason decision-making approaches have been widely adopted in trust evaluation. The decision support system is a typical solution to assess trust. It collects and filters out considerable data, and makes decisions. Multiple criteria decision-making (MCDM) or multiple attributes decision analysis (MADA) is the critical method for the decision-making approach. Group consensus-based evidential reasoning (GCER) [76] is an approach that assigns weights to attributes specified by the stakeholders as part of MADA.

General trust perception model (GTPM) [71], the main idea is to have a set of trust factors that include those already presented in the literature and to adapt the model to a specific domain by changing the weight of each factor. It contains nine factors: security, privacy, brand name, risk, third-party seals, usability, reputation, reliability & availability, and look & feel, which refers to the correlation analysis to the domains of e-banking, e-commerce, e-health, and e-portfolio. Additionally, there are evaluation tools that attempt to reduce the subjectivity of trust, such as the approach proposed by Zhang et al. (2004) which attempts to address subjectivity in the method, including personality to fix subjective, information collection, trust decisions, and inputs of trust functions [77]. Blockchain-based decentralized trust evaluation methods [50] and natural language processing (NLP) [78], on the other hand, collect and analyze large amounts of data in order to compensate for subjectivity or bias.

2.5 | Practice in Software Evaluation

The essence of software selection and evaluation is the judgment of a series of information raised by the software-producing organization within SECO. In practice, a difficult challenge is that the information provided by the software providers is not sufficient to help the trust information consumers make a judgment [58]. Often software stakeholders gather trust facts from multiple platforms.

GitHub is a web-based git repository, that showcases source code, project documents, and development activities, such as issues and commits, as well as statistics about software components, such as the number of forks, watchers, contributors, downloads, stars, and open issues [79]. These facts include software components, software engineers, community, end-users, and dependencies. The statistics are usually influenced by the level of interest that software development and selection naturally have in a product, which means the more popular a project is, the more trustworthy it is [80]. Therefore, GitHub has become the most popular code-sharing platform for providing information and statistics to an increasing number of software end-users and engineers to help them determine whether a software product is trustworthy.

Libraries.io [†] is another web-based platform that monitors the release of packages, provides code, communities, distribution, and documentation for each project, and maps the dependencies between packages. Thus, it can provide the number of projects dependent on the package and the number of repositories dependent on the package directly. Additionally, Libraries.io provides an algorithm for indexing search results, SourceRank. It directly shows the quality and trustworthiness of a package, which is based on a set of package attributes to facilitate the comparison of different packages [81]. The attributes that measure the use of best practices include code, community, distribution, documentation, and usage.

Trusted Product Maintainability ISO/IEC 25010 Quality Model supporter by **TIOBE / TÜViT**[‡] is a computational and qualification model for software maintenance based on the ISO/IEC 25010 standard about software product quality. It is at the source code level, calculating the quality score of commercial software code based on five metrics, namely cyclomatic complexity, compiler warnings, Coding standards compliance, code duplication, and fan out. Scores are calculated separately for each metric, and the final score is calculated as the average of the five scores [82].

[†] https://libraries.io/
[‡] https://www.tiobe.com/tiobe-index/

Snyk.io[§] is a well-known and widely used platform in the field of software security and vulnerability management. The Vulnerability Scanner is designed as a developer security solution to strengthen the security of code, dependencies, containers, and infrastructure represented by code. The advisor function provides health scores for and compares a wide range of open-source packages, including NPM, PyPi, Go, and Docker. It shows health-related information from the perspective of popularity, security issues, maintenance, and community.

With the transparency and availability of open source software (OSS), a number of researchers or organizations have proposed various approaches or tools to help decision-makers select a more trustworthy OSS. However, most of them are presently not supported. **OpenBRR** is an assessment framework, that comprehensively evaluates the facts of role, Functionality, Usability, Quality, Security, Performance, Scalability, Architecture, Support, Documentation, Adoption, Community, and Professionalism of an OSS product. **QSOS**[¶], is a free project intended to improve technological awareness of open-source components and projects. It proposes a method for qualifying, selecting, and comparing open-source solutions on the project's maturity and the software's functional coverage. **CHAOSS**[#] an initiative under the Linux Foundation, is dedicated to the development of metrics, metric models, and software aimed at enhancing our comprehension of the well-being and dynamics of open-source communities on a worldwide scale. CHAOSS stands for "Community Health Analytics in Open Source Software". The CHAOSS software collects and visualizes OS community health data. The 88 metrics cover a wide range of quantitative measures, such as SECO, software, contributors, and communities.

3 | RESEARCH METHOD

We performed the systematic mappings following the guidelines and steps of Kitchenham[83] to research the effective approaches to managing software trust. The main research question is: **What are the research challenges faced in the domain of software ecosystem trust evaluation?** with the following research questions (RQ) in Table 2. The study only includes the manuscripts presenting concrete trust evaluation approaches in the context of software selection. The protocol is available as online material [‖]. The manuscript selection process is shown in Figure 2.

TABLE 2 Research Question and Motivation

No.	Research Question	Motivation
RQ1	What are software ecosystem trust evaluation approaches?	To explore the methodologies, tools, or techniques used to assist the trust evaluation in SECOs. These approaches evaluate the overall trust or trust facts of the various components, entities, and interactions within or across SECO(s).
RQ2	How is the software ecosystem trust evaluation defined?	To explore aspects or trust factors considered in software trust assessments in different contexts.
RQ3	How is the software ecosystem trust evaluation result presented?	To explore the presentation of software trust evaluation results. The results may vary depending on the specific evaluation framework and the target audience, e.g., trust score, visual Representations, or only positive or negative.
RQ4	Which entities of the software ecosystem have been covered in the approaches?	To explore the coverage of SECO entities, e.g., software components, software providers, or software services. It enables us to gain insight into the completeness of the SECO trust evaluation.

We adopted Publish or Perish 8 for the manuscript selection to perform an automatic search. We followed the PICOC (Population, Intervention, Comparison, Outcomes, and Context) to define the search string[84]. For this study, the population in this search string refers to entities or elements within the context of software ecosystems, including software, components, packages, or any SECO entities. The intervention involves actions such as selection, evaluation, and calculation of trust in software or its components. The comparison involves contrasting different models, frameworks, techniques, tools, approaches, processes, criteria, methods, or methodologies used for trust-related activities in SECO. The outcomes are related to the results or implications of trust-related interventions in software ecosystems. The context considers various aspects related to trust in software and its components. The search string is ***trust and (software or component or package or software ecosystem***

[§] https://snyk.io/
[¶] https://www.qsos.org/
[#] https://chaoss.community/
[‖] SM protocol and extracted data

or software ecosystem entities) and (selection or evaluation or calculation) and (model or framework or technique or tool or approach or process or criteria or method or methodology). The sources include Wiley Online Library, Springer, ACM, ScienceDirect, IEEE Xplore, Taylor & Francis, Emerald Insight, and Cambridge University Press. Following the application of inclusion and exclusion criteria, we found 44 manuscripts, of which the majority are for common trust evaluation, or trust in areas other than SECO, such as trust in cloud computing, or do not provide specific solutions. We adopted another round of searching by using Elicit: The AI Research Assistant**, with the research question "What are software ecosystem trust evaluation approaches?" All searches were finished by 23rd June 2023. In addition to the 100 extracted manuscripts, 39 manuscripts meet the quality assessment criteria. The sources of the manuscripts include IEEE Xplore, Springer, MDPI, ScienceDirect, ACM, Academy publisher, Hindawi, NADIA, RAMS Consultants, Taiwan Academic Network Management Committee, and Taylor & Francis. Finally, we got 42 selected manuscripts.

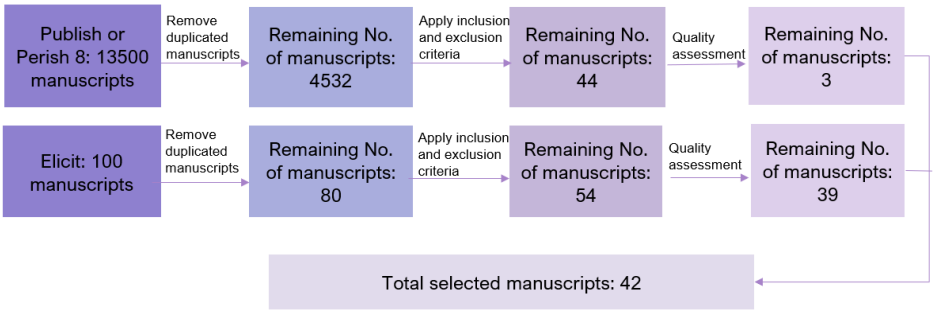

FIGURE 2 *This figure indicates the number of manuscripts at each stage, we show the publisher of the final selected manuscripts*

We consider the **validity** of the following points. First, our selection of manuscripts was inevitably subjective due to bias and understanding of the concepts, so we could not have included all relevant studies in this SM. Second, our search was built in the context of software selection or evaluation in the context of SECOs, so not all SECOs were included, e.g., mobile software ecosystem, cloud ecosystem, and IoT ecosystem. Moreover, we incorporated specialized tools to enhance and expedite the data extraction process. One such tool, Elicit, automates the snowballing technique, streamlining the search process. However, it may cause the search results to be skewed towards particular topics. Finally, most of the models have not yet been widely adopted, which means that they still lack validation and application in real projects and may be problematic in terms of generalizability.

4 | RESULTS

This section reports results for each research question based on 42 selected manuscripts. The selected manuscripts are primarily from IEEE Xplore and Springer and range from 2007 to 2022. All the selected manuscripts present concrete solutions for SECO trust evaluation approach. The validation methodology of these approaches consists of 24 case studies, seven experiments, seven unvalidated studies, two numerical simulations and theoretical analyses, one validity test, and one comparative study.

4.1 | RQ1-What are software ecosystem trust evaluation approaches?

4.1.1 | Evidence-based approach

The software trustworthiness evaluation approach as *"the procedure of determining the trustworthy degree of a software program with given trustworthy degrees of attributes"*[28]. Trust attributes usually correspond to a set of trust facts. The evidence-based approach is one of the most utilized methods from the selected manuscripts. It is trust evidence or facts-driven methodology for assessing and making trust-related decisions. Typically, it defines trustworthiness attributes, assigns weights to those attributes, and then uses some formula to calculate a trust score for the software.

** https://elicit.com/?workflow=table-of-papers

TABLE 3 Summary of manuscripts based on evaluation techniques and entities of SECO (order by year)

No.	Ref	Year	Technique	Entities
1	22	2007	Trust certificates based on specific attribute profile	Software service
2	10	2007	RAP method	Open-source components
3	34	2008	N/A	Component software system entities
4	32	2008	DS theory + Entropy	Software product
5	7	2008	Trusted software theory	Software product
6	9	2009	AHP + Multi-attributes group decision-making	BPM software
7	38	2009	Process management	Software process
8	42	2009	Axiomatic approach	Software product
9	18	2009	FAHP + TOPSIS	Software product
10	15	2009	Distrustable factors analysis	Software product
11	35	2010	FCMs	Component-based software systems focusing mainly on system runtime
12	20	2010	Entropy	Software (risk)
13	12	2010	Exponential smoothing technique	Software service (SaaS)
14	2	2010	Analysis method based on rules	Software product
15	14	2010	MCDA	Software product
16	3	2010	AHP	Software product
17	21	2010	Fuzzy Grey Method	Software product
18	1	2011	Sugeno fuzzy inference, fuzzy set theory	Infrastructure as a service (IaaS)
19	36	2011	Classification evaluation model	Software product
20	6	2012	ER based STE method	Software based on the uncertain and unreliable environment
21	40	2012	Axiomatic approach	Large-scale software system
22	19	2012	FAHP + TOPSIS	Software product
23	5	2012	ER based STE method	Software product
24	41	2012	Fuzzy comprehensive evaluation model	Software product
25	16	2012	Finite state machine	Software product
26	33	2013	Fuzzy set theory	Software product
27	39	2013	Axiomatic approach	Software product
28	17	2013	Behavioristic model	Software product
29	11	2014	POME + GDMM	Software architectures
30	29	2016	Axiomatic approach	Software product
31	8	2017	AHP	Web application based on software product
32	30	2018	Axiomatic approach	Source codes
33	37	2018	Fuzzy set theory	Software product
34	26	2019	Axiomatic approach	Software product
35	4	2019	Axiomatic approach	Software product
36	28	2019	Axiomatic approach	Software product
37	31	2020	Deep learning	Software behaviour
38	23	2022	FAHP + TOPSIS	Software product
39	13	2022	AHP	Software product
40	24	2022	Fuzzy set theory	Software product
41	27	2022	Axiomatic approach	Software product
42	25	2022	Axiomatic approach	Software product

Immonen et al. (2007) present a method to evaluate the direct and indirect trustworthiness of open-source components, architecture, and systems. The direct evaluation contains the technical evaluation by facilitating both reliability analysis and testing. The indirect evaluation consists of non-technical evaluation, such as component reputation, component history, and license. These tools are designed to operate seamlessly within the Eclipse environment, promoting interoperability with a range of other software development tools. Notably, the tools employ a Markov chain model to compute Probability of Failure (PoF) values for individual components. At the system level, the tools take sequence diagrams and simulation models as inputs, simulate system behavior at the architectural level, and calculate PoF values for each component within the target environment. This comprehensive approach provides insights into the reliabilities of system execution paths, contributing to overall trustworthiness assessment [10]. Zheng (2008) introduces a trust management approach based on trust evaluation, both at component download time and runtime. The focus is on opinions related to quality attributes from different entities. To ensure the integrity of a component, verification and validation of its origin, information, and integrity are essential before downloading. This is achieved through the component certificate verification mechanism. Runtime integrity is maintained through performance evaluation of the component [34]. The model proposed by Li et al. (2009) focuses on identifying distrustable factors and controlling them to achieve a trustworthy software state. While the paper provides insights into identifying factors that erode trustworthiness, it does not delve into the specific calculation of trustworthiness values [15]. A multi-dimensional and multi-scale trustworthiness

evaluation index system is established by Yang (2011) for software of the same type within an XML database. This system guides trust evaluation for other software products of a similar type. Weight assignments for each trustworthiness attribute are determined based on their impact on related trustworthiness expectations. Trust factors include efficiency, complexity, maintainability, usability, reliability, security, ease of use, reusability, real-time capabilities, survivability, and portability[36]. Ding et al. (2012) propose a weight-based evidential reasoning approach that employs a total uncertainty measure in order to solve Software Trustworthiness Evaluation (STE) problems with specific trustworthiness requirements[5]. The trust attributes are self-defined, including, for instance, reliability, safety, real-time, maintainability, availability, and security, which are derived from ISO 9126. Incorporating a measure of total uncertainty into the evidential reasoning (ER) approach, this study establishes a mathematical model for the objective determination of weights. Similarly, Ding et al. (2012) enhance the ER-based STE method and propose two discounting factor estimation approaches to meet the demands of the unreliable trustworthiness evaluation pretreatment[6]. Discounting factor is a concept that reflects the degree of evaluation reliability. The utility theory which is based on Trustworthiness Evaluation Distance (TED) is employed to characterize the degree of evaluation reliability.

Numerous studies have approached trust assessment as a decision-making problem. In STE, decision-making systems frequently approach the problem as MCDM or MCDA challenge. Fuzzy Analytic Hierarchy Process (FAHP), Analytic Hierarchy Process (AHP), and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), fuzzy set logic are common techniques adopted in the solution. Shi and Yang (2009) utilize the FAHP method to establish weights for evaluation criteria within a hierarchical structure. They apply the FTOPSIS method to rank software alternatives. Additionally, they employ fuzzy triangular numbers to address uncertainty and vagueness in the evaluation process[18]. Alhamad et al. (2011) propose a trust evaluation approach for cloud applications but it can be extended to various web-based applications. The approach allows the requestor to set up a fuzzy set, trust factors, and related weights. It uses the Sugeno fuzzy technique with the Gbell membership function to calculate the trust degree of the application[1]. Shi et al. (2012) propose an approach for assessing software trustworthiness based on combination weights (CW) to improve the TOPSIS method. The key evaluation criteria can be derived from comprehensive investigation and consultation with different experts from diverse fields. According to expert judgment, the FAHP method is used to determine the importance of the degree of criteria. Objective weights are calculated objectively using the entropy weighting method in order to avoid subjective one-sidedness[19]. Zhang et al. (2012) introduce an evaluation index set encompassing five generally accepted attributes of trustworthiness. A fuzzy composite evaluation model is utilized to assess these software trustworthiness attributes. Experts are responsible for assigning weights to two-grade indexes[41]. The research proposed by Wang et al. (2013) emphasizes the role of evidence during software usage, encompassing QoS information in runtime and user feedback. It extends to allow experts to define specific trust requirements. Factors within the Trust Requirement Model encompass satisfaction, security, usability, performance, and reliability (availability, fault tolerance), with a foundation in fuzzy sets[33]. Yang et al. (2018) introduce a unified software trustworthiness framework based on trust theory from humanities and sociology[37]. This framework encompasses a range of trustworthiness measurement methods, including the advanced J-M model for ability trustworthiness measurement, a comprehensive evaluation method based on fuzzy set theory accounting for multiple attributes, and identity trustworthiness measurement based on code homology detection tools. The model includes elements such as initial trust (requirement), knowledge-based trust (correctness, security, reliability, basic standard, identity), field regulation, and standard authoritative evaluation agencies. Trust factors are derived from IEEE Std 610-12-1990, with weight assignments tailored to software requirements. Similarly, Tao et al. (2022) propose a model based on combination weights (CW) to improve TOPSIS methods. The weights include subjective weights and objective weights. FAHP has been adopted to determine subjective weights, Entropy is used to determine objective weights[23].

AHP was proposed by Dr. Thomas Saaty in the late 1970s and has been applied in a wide variety of applications in various fields. This method allows consideration of both objective and subjective factors in selecting the best alternative[85]. It has been widely adopted in the trust evaluation approach, especially in analyzing the weightings within the approach. Hou et al. (2009) introduce a set of linguistic evaluation scales and related concepts, proposing a multi-attribute group decision-making model based on AHP and linguistic scales[9]. This model considers trust factors such as usability, reliability, security, integrity, testability, and maintainability. Weight assignment is achieved through the AHP method, with expert decision-making adhering to the One-vote veto principle. Chen et al. (2010) propose an evaluation process model for software trustworthiness[3]. This model incorporates a range of factors, such as availability, reliability, and security. Weight coefficients for these factors are determined through subjective assignment, drawing on the expertise of professionals and assessment experts. Additionally, objective assignment methods such as the AHP and fuzzy neural network techniques are employed to determine weight coefficients. A trust evaluation model is introduced by Song and Wang (2010), leveraging a fuzzy grey comprehensive evaluation method. This model is tailored to assess the trustworthiness of software systems. Weight assignment for each evaluation index is accomplished

using the AHP[21]. Han et al. (2017) present a model centered around web applications and encompassing factors such as interface, documentation, design, code, and product. Detailed information on these factors, including sub-factors, is provided across five tables[8]. In a recent study, Liu et al. (2022) employ the AHP to integrate measurement values. AHP facilitates the decomposition of complex multi-objective decision problems into sub-objectives and multiple levels of indicators. These weights were determined through expert decision-making and the weighted sum method. The trust factors include characteristics from ISO/IEC 25010 standards, general requirements from ASPICE, and the evaluation process outlined in ISO/IEC 25040. These diverse methodologies contribute significantly to the advancement of software trustworthiness assessment[13].

There are some other decision-making techniques. Based on the aggregation procedures, MCDA methods are classified into two categories: compensatory and noncompensatory. The weighted average sum (WAS) aggregation technique and the techniques based on multi-attribute utility theory are compensatory[14]. Lu et al. (2010) adopt ELECTRE TRI, a noncompensatory MCDA method for the software trust evaluation. In this evaluation process evaluators need to define the problem situation of trustworthiness evaluation, and based on the problem define the formulation, determine the evaluation model and its parameters; then apply the evaluation model[14]. Jiang (2014) introduces the Principle of Maximum Entropy (POME) and Grey Decision-making Method (GDMM) as trustworthiness evaluation techniques for software architecture. Their work not only utilizes these methods but also demonstrates their scientific validity and rationality[11].

4.1.2 | Axiomatic approach

A novel method for assessing software trustworthiness is introduced by Zheng (2009). This method relies on statistical indices and is anchored in the domain of pure mathematics, offering a rigorous and analytical approach[42]. Song et al. (2010) utilize Entropy to uncover the reasons for software uncertainties and provide the mechanism of risk management in the development life cycle. Information entropy is introduced to assess risk uncertainty and a three-dimension risk assessment was obtained through a questionnaire survey. Risk management factors include requirement risk, user risk, developer risk, project management risk, development risk, and environment risk[20]. Zhang et al. (2013) present an extension of Tao's five measurement models of software trustworthiness, grounded in software attributes[39]. The approach utilizes axiomatic methods and entropy calculations to assess software trustworthiness. An axiomatic approach is adopted to validate the developed measure of software trustworthiness which is based on extensive structure in measurement theory. The theory is applied to the design of a source code-oriented software trustworthiness measure based on an extensive structure in measurement theory[29]. Later, the authors enhance the properties from four to seven and develop an improved software trustworthiness measure by partitioning critical attributes into several groups[26,25,28]. Trust attributes within this axiomatic approach primarily concentrate on the Software Development Life Cycle (SDLC). Statistical techniques have been adopted to evaluate the trustworthiness as well, for instance, structural equation modeling (SEM). Deng et al. (2019) employ an SEM-based method to assess the software trustworthiness. Initially, the authors create a comprehensive indicator system for evaluating software trustworthiness. Utilizing survey data, authors develop an SEM model for evaluating software trustworthiness, enabling us to determine attribute weights and their interrelationships. Finally, they employ the trustworthiness measurement model to compute trustworthiness scores for the surveyed software[4]. In the work of[27,30], comprehensive efforts are made to identify critical attributes and sub-attributes crucial to software trustworthiness, with a particular emphasis on quality attributes. Experts play a central role in this attribute identification process, employing Saaty's 9-point scale to assign weights based on their discernment and expertise.

4.1.3 | Others

According to Yang et al. (2009), the Trustworthy Process Management Framework (TPMF) toolkit is designed to support measuring and improving process trustworthiness in order to assess and assure software trustworthiness during process runtime[38]. Limam et al. (2010) contribute by proposing an automated rating and reputation computation model for reputation-aware software service selection. It explores the process of integrating quality, cost, and reputation into decision-making. Specifically, it considers the temporal aspect of feedback, acknowledging that feedback trends are useful for identifying fluctuations in service quality[12]. According to Yan et al. (2010), trustee quality attributes are evaluated using trustor criteria. Various trust control modes are introduced in order to manage and enhance entity quality attributes, taking into account trust relationships within different runtime environments, such as component, system, and execution environments. While the authors employ fuzzy cognitive maps (FCMs) to illustrate relationships among trust factors, evaluation criteria, and trust control modes, this approach is not directly used to make trust decisions[35]. Wang et al. (2010) propose a method for assessing software trustworthiness classification. The

method collects trust evidence throughout the SDLC and accumulates it through Dempster-Shafer (DS) theory to measure the trust status of each attribute while using entropy as an uncertainty factor for the evidence. DS theory is one of the main tools for reasoning about uncertain information data obtained from multiple sources[32]. Nami et al. (2012) propose a model centered around interaction scenarios between software, users, and the environment. This model relies on expert-generated scenarios and user cases and testing to assess software trustworthiness[16]. Tian and Guo (2020) present a model that leverages the behavior trajectory matrix technique to evaluate software trustworthiness. This model classifies software behavior by converting behavior trajectory matrices into grayscale images and employing a deep Residual Network (ResNet). The cosine similarity algorithm then assesses deviations from normal behavior trends, resulting in a dual evaluation of software trust[31].

Insight of RQ1: The SECO trust evaluation approach tends to start from the essence or concept of trust, i.e., based on a series of data predictions. The more information available, the more accurate the trust assessment. Some methods are based on evidence-driven approaches. Trust evaluation is usually translated into a multi-criteria decision problem. The weights of trust factors are calculated by AHP or FAHP and candidate alternatives are ranked by TOPSIS. There are also methods that employ other techniques for obtaining evidence of trust or presenting trust results. One trend is that post-2016 research is actively exploring the use of axiomatic approaches to trust problems. Researchers address software trust as a mathematical problem rather than from a software engineering or SECO perspective. In addition, the approaches from the selected manuscripts are theoretical or academic. Even though the acquisition of trust factors, and their weighting assignments come from experts or users of the approaches, the ease of use of the approaches is not taken into account. Generally, these methods are complex and require people with extensive knowledge of maths and decision-making systems to use them.

4.2 | RQ2-How is software ecosystem trust evaluation defined?

Unfortunately, we could not find a definition for software ecosystem trust evaluation. At least, based on the search string and research questions, we cannot get a definition from the result. Drawing upon the definition of SECO, there are three major elements: *common software*, *business*, and *connecting relationship*[44]. In SECOs, actors assume a pivotal role, serving as a foundational component for two of the three identified elements. The emphasis on connecting relationships underscores users or actors in the SECOs. Because connecting relationships are a set of users or actors in the ecosystem, the business is the community of users that should be satisfied. However, the SM result reveals that the trust of SECO fully relies on the trustworthiness or quality of software, without considering the impact of actors.

Evidence-based approaches are the most common. This approach relies on a systematic analysis of trust factors and their corresponding weights. Trust factors are expressed in the selected manuscripts as trust attributes, trust indicators, or trustworthiness attributes. Figure 3 highlights a dominant emphasis on trust factors related to software quality and security from the selected manuscripts. Maintainability, safety, and availability are important considerations as well. Critical entities, software-producing organizations, and software engineers have not been fully evaluated, with only limited consideration of the related factors of reputation, support, and service. These trust factors are derived from the following sources: (1) from established standards including Capability Maturity Model Integration (CMMI)[38], IEEE Std 610-12-1990[37], ISO/IEC 14598[3], ISO/IEC 9126[5], ISO/IEC 25010[13], ISO/IEC 25040[13], and ISO/IEC 25051[13]; (2) trust factors are established through a predefined by existing literature; (3) gathered from interviews or surveys; and (4) identify by the approach users, including experts, decision-makers, and trustors. For the first three sources, the trust factors remain static and unchanging. For the last one, trust factors are dynamic. The adaptability of these dynamic trust factors reflects the subjective nature of trust assessment, through the experience and expertise of the users to refine the trust assessment process to meet the user requirements, context, and specific environment.

The determination of weight in trust factor evaluation typically falls into two categories: subjective and objective. Subjective weights are assigned by experts, decision-makers, and trustors based on their domain knowledge and expertise. In contrast, objective weights are calculated automatically using mathematical models, without any subjective influence[19]. Most approaches take a manual approach to assigning weights, and only some take a combination of the two.

Insight of RQ2: The studies do not provide a definition of SECO trust evaluation. Software quality is the primary determinant of trust. While the reference to trust factors matches the frequencies observed in literature-based trust factors,

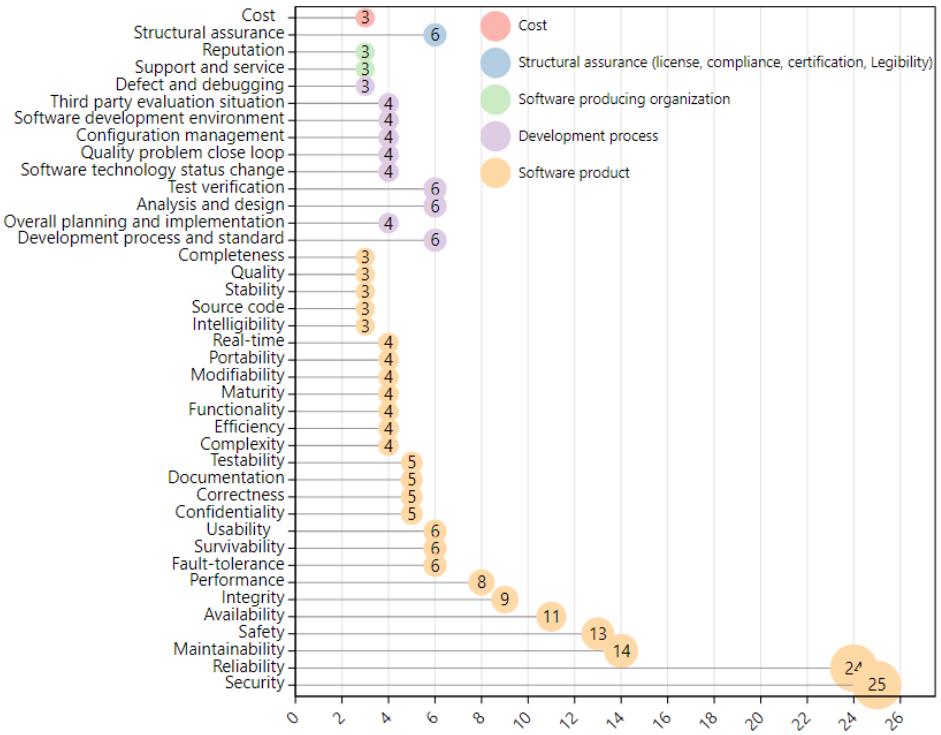

FIGURE 3 *This figure provides a summary of manuscripts based on the number of trust factors*

most of these factors are related to software products, including software quality and development processes. There are still gaps in the coverage of structural assurance, cost considerations, and software-producing organizations. Other key SECO entities, such as software packages, package managers, and product categories are not covered. Additionally, manual assignment of weights is the dominant method. These limitations highlight the need for a broader, more objective, and comprehensive trust evaluation approach to cover a wide range of SECO entities to avoid a bias in the results.

4.3 | RQ3-How is the software ecosystem trust evaluation result presented

As indicated in Table 4, most evaluation results are expressed numerically. Approximately one-third of the approaches present their results in the form of a trust score. In nearly all the result-display formats utilizing numerical values, those numbers are equal to or greater than 0. This trend aligns with the belief held by researchers that non-negativity is a characteristic associated with software trust[27]. The most common format ranges from 0 to 1 or 0 to 10, with trust scores rarely exceeding 10.

Seven approaches use degree-based representations, categorizing trust into multiple degrees (e.g., three to five levels) corresponding to different trust scales. Trust scores are calculated using trust evaluation models, and the trust degree is determined based on the scale on which the score falls.

Another method of trust evaluation involves decision-making systems that utilize predefined rules to make trustworthiness assessments. For example, a trust degree falling between 0.75 and 0.85 may result in a “Positive trustworthy” trust level, with the decision-making rule being "Deploy after local optimization" [5]. Some approaches solely make binary decisions, categorizing software as trusted or distrusted.

There are other forms of presenting results, such as candidate rankings, trust analysis reports, and graphical evaluations. Notably, some approaches do not directly display the trust score of the software but instead focus on the trust score of specific software sub-attributes or related dimensions[13,20].

TABLE 4 Summary of manuscripts based on evaluation result format

Result Type	Format	Ref
Number	0-1	13,30,29,9,14,15
	0-10	27,39,25,26,4
	1-10	23,24
	0-1 It shows the trustworthiness score for sub-attributes.	13
	0,1,3,5,7,9 If the user hesitates between two adjacent scores, then scores it one of 2, 4, 6, 8	14
	Numerical, the unit is K$. It is a loss indicator to unify three parts of trustworthiness result, hence if the value is smaller, the trustworthiness of software will be higher. If it is bigger, the trustworthiness will be lower.	37
Degree	Very poor (VP), poor (P), good (G), very good (VG), and excellent (E) It is based on fuzzy rules of input trust factors (e.g., scalability, availability, security, and usability) and fuzzy sets (e.g., low (L), medium (M) and high (H)).	1
	A (value of indication > 0.95), B (0.85 < value of indication $\leq$ 0.95), C (0.7 < value of indication $\leq$ 0.85), D (0.5 < value of indication $\leq$ 0.7), and E (value of indication $\leq$ 0.5)	3
	High trustworthy (0-0.25), trustworthy (0.25-0.5), low trustworthy (0.5-0.75), untrustworthy (0.75-1)	19
	Low, medium, High Calculation is based on a comprehensive evaluation model.	41
	High credible (V1), Average credible (V2), Low credible (V3), Incredible (V4) Calculation is based on four white weight functions.	21
	Excellent (90-100), good (75-90), medium (60-75), fail (<60)	8
	Trust, basic trust and distrust Calculation is based on the floor and ceiling of the expected value of the user.	33
	It calculates the trustworthy degree of software sub-attribute, including Rank V (0.95-1) Rank IV (0.85-0.95) Rank III (0.7-0.85) Rank II (0.45-0.7) Rank I (0-0.45). The calculation is based on a mathematical programming approach.	28
	Rank0: the trustworthiness is unknown. Rank1: the assessed software is usable. Rank2: the assessed software has variable trustworthiness attributes. Rank3: the assessed software has verifiable successful cases. Rank4: the trustworthiness of the assessed software passed verification and analysis of independent authorities. Rank5: the trustworthiness of the assessed software is provable.	32
	T0 (unknown number of trust principles met) to the highest rating T5 (all trust principles met).	38
Decision	1) If the software behavior is in an abnormal state, the administrator is notified by the display or via an alarm to suspend operation. 2) If the software behavior is in a normal state, the administrator can be notified by the display or via an alarm to pay attention to it, or no adjustment will be made. Decision is made by a software trustworthiness evaluation model based on a behavior trajectory matrix, namely, BTBM-TM.	31
	Trust, distrust Decision is made by specific trust evaluation models	34,35
	Refused to deploy-Completely untrustworthy [0, 0.3], Re-evaluation after substantial modification-Positive untrustworthy (0.3, 0.55), Re-evaluation after minor modification-Slightly untrustworthy (0.55, 0.65], Deploy after global modification-Slightly trustworthy(0.65, 0.75], Deploy after local optimization-Slightly trustworthy (0.75, 0.85], Risk-free of application deploy-Significantly positive trustworthy (0.85, 0.95], Deploy with no changes-Completely trustworthy (0.95, 1] Calculation is based on objective weight-based ER (OWER) method and utility function.	5
	H1-Completely untrustworthy, refused to deploy; H2-Untrustworthy in a large proportion, re-evaluation after deep modification; H3-Average level of trustworthy, deploy after local modification H4-higher level of trustworthiness, deploy after local optimization; H5-completely trustworthy, risk-free of application, deploy. Calculation is based on discounting factor estimation approaches and utility functions.	6
Others	Trustworthy analyzing report based on software and related resources	2
	Graphical view based on direct evaluation for the technical evaluation and component testing, indirect evaluation for non-technical evaluation, and technical trustworthiness evaluation for technical trustworthiness evaluation.	10
	By comparing Real Behavior with Reference Behavior in statechart to identify the behaviors which hurt software trustworthiness.	16
	It shows three-dimensional risks (probability, uncertainty, and loss) for all risk attributes by a number from 0-1 for risk factors in the software lifecycle. The calculation is based on the information entropy approach.	20
	a. The calculation is based on an evaluation approach using the FAHP to determine the weights of evaluation criteria and utilizing fuzzy TOPSIS to obtain the final ranking of the software alternatives. b. The calculation is based on the Principle of Maximum Entropy (POME) and the Grey Decision-Making Method (GDMM).	18,11
No evaluation result or example presented in the literature		36,40,22,17,42

Insight of RQ3: These results demonstrate the various ways in which trust evaluation results can be presented. A trust result is presented numerically and degree-based as a visual indication of the level of trust, while approaches in the form of decisions are related to the subsequent actions that are being taken. Other forms of results focus on specific sub-attributes or related dimensions of trust. Different ways of presenting trust evaluation results serve different purposes, requirements, and preferences of trust evaluation.

4.4 | RQ4-Which entities of the software ecosystem have been covered in the approaches?

According to Table 3, of the 42 manuscripts, 28 are centered around software and software products, and 2 are dedicated to software services. It is worth noting that of these 28 contributions, almost all of them explicitly stress the concern for software trustworthiness in their titles. In addition, there are a number of approaches that address aspects of software components, covering a range of topics such as business process management (BPM) software, cloud services, web applications, runtime aspects of component-based software systems, entities within these systems, large-scale software systems, software operating in uncertain or unreliable environments, OSS, reusable software components, software architecture, software behavior, software process management, software risk assessment, and source code analysis.

Insight of RQ4: Most of the selected manuscripts relate to software, software products, and software components, with a few manuscripts focusing on software services, operating systems, software architecture, software process management, and source code. While our search string in the SM protocol included "software ecosystems" and "software ecosystem entities", the search results and quality assessment results were not satisfactory, leaving space for future work[54].

In current trust approaches, there appears to be a significant gap in terms of fully taking into account other SECO entities such as software packages, package management systems, application stores, and software-producing organizations[54,86]. Also, these approaches fail to take into account the attributes of other SECO entities, such as trust of the software-producing organizations, trust in brands, and trust in product categories. Consequently, current approaches do not provide a complete picture of SECO trust; rather, they are more of an evaluation of software quality.

5 | DISCUSSION

The integration of information technology into every aspect of daily life, from communications to commerce to education, is a monumental shift of historic proportions. However, while enjoying the convenience brought about by information technology, we also have to face a series of troubles. Data breaches, cyber attacks, frequent software bugs, system failures, and unreliable services reduce people's trust in software products. Can we trust the software products we use? That is the question that individuals and organizations are asking all the time. Our findings of the RQs show that SECO trust evaluation approaches are not yet mature enough with experimental results. The discussed frameworks involve technical assessments such as reliability analysis and testing, as well as non-technical evaluations like reputation and licensing. Software trust evaluation is generally seen as a decision-making challenge, employing methods such as FAHP, AHP, and TOPSIS. These approaches are underscored by the importance of both subjective and objective factors in the evaluation process. Additionally, we discuss the use of compensatory and noncompensatory decision-making techniques in trust assessment, reflecting its critical importance in the software industry. Although a number of researchers have proposed sophisticated methods for integrating all of the individual attributes of software trustworthiness, it is a difficult task in a systematic manner[41]. Most trust calculations are converted into decision problems or mathematical problems. It requires the evaluator to have more specialized knowledge to understand and use them. And the results of evaluations are difficult to validate or compare. Additionally, software is not the only factor that impacts SECO's trust. SECO is majorly based on "actors". In contrast, actors are rarely considered in approaches. Approaches from the literature have focused more on the quality of software products. The existing popular approaches focus more on source code and the security of OSS. Only GitHub and Libraries.io provide trust information about the community, e.g., GitHub star, the number of contributors, and the number of subscribers. Based on the findings, we identify the following outstanding research challenges.

5.1 | Research Challenges

Lack of trust in SECO entities: SECO trust constitutes both upstream and downstream trust between the stakeholders that comprise the SECO[48], for instance, trust between software end-users and software products, trust between software end-users and software-producing organizations, trust between software engineers and third-party software components, and trust between software engineers and software package managers. Currently, most approaches mainly focus on the trust evaluation of software, software products, and software components, with a few manuscripts focusing on software services, operating systems, software architecture, software process management, and source code, without considering the trust of other entities as well as related attributes, e.g., trust between software engineers and software providing organizations. Failure to consider trust between SECO entities creates a cognitive bias that prevents a comprehensive and objective understanding of SECO health. In other words, the challenge lies in expanding the scope of trust evidence or attributes beyond software quality attributes[17].

Lack of generic method for trust evaluation: One prominent challenge lies in the absence of a generic method for assessing SECO trust[87]. Trust is inherently multidimensional, necessitating the consideration of multiple attributes during the selection process[63]. Although trust information consumers have different considerations, they still need to get the full picture when it comes to evaluating trust[50]. The lack of a generic trust evaluation approach is twofold. (1) There is a lack of integrated and comprehensive trust facts. When a large number of trust facts are collected into a unified solution, it is a challenge to filter, identify, analyze, and summarise the overload of information into real trust facts. (2) SECO covers a wide range of scenarios, and different ecosystems involve inconsistent trust facts. For example, the selection of OSS is more about technology, while the selection of closed software pays more attention to the mature and stable products, as well as the certifications, reputations, and services provided by the software providers. Additionally, each SECO may have its own unique characteristics, stakeholders, and trust requirements, and each stakeholder group may have its priorities and interests that influence the trust decision-making process. The outcome of trust may not meet the requirements of all stakeholders. Therefore figuring out the evaluation result is another challenge.

Subjectivity in trust evaluation: Trust is inherently subjective and can vary based on individuals' experiences, biases, and education[88,56]. The pursuit of trustworthiness in software behavior necessitates minimizing the influence of human factors and ensuring the objectivity of the evaluation process. This objective underscores the need for robust and impartial evaluation methods[31]. It is common for current evidence-based approaches to rely on subjective methods for identifying trust factors and assigning weights. This subjectivity introduces a degree of uncertainty into the assessment process. Therefore, it is necessary to minimize human interference to ensure objectivity in the assessment of software behavioral credibility[31].

Excessive environmental impact due to complexity: In dynamic environments, a variety of contextual factors may change and affect trust evaluation, it is particularly challenging to develop mechanisms that can effectively detect these contextual factors[22]. There is a lack of effective correlation between observations and their context or environment in current approaches. Software performance can be negatively affected by, for instance, network congestion, energy consumption, and server outages. For this reason, it is necessary to remove or reduce the weight given to these factors in the evaluation process to balance the impact of environmental stress on software performance or services. Obtaining runtime operational data may also be a solution.

Lack of risk contextualization in software supply chain: Trust must be accompanied by a degree of risk and vulnerability. There is little consideration of risk assessment in current approaches. A risk assessment of trust evaluation approaches can help identify vulnerabilities and potential weaknesses within SECO entities, enhance trustworthiness, and verify that trust factors effectively mitigate risk and ensure trust at the required level.

Attacks in trust evaluation approaches: Typically, trust evaluation techniques focus solely on the collection of trust facts, or trust calculations, ignoring other aspects of security such as privacy protection and attack resistance. In the 42 manuscripts we reviewed, we did not find any relevant information. However, a series of attacks have been launched specifically targeting trust evaluation tools and reputation systems, including Bad-mouthing attacks, On-off attacks, Conflict behavior attacks, Collusion attacks, Whitewashing attacks[87], Sybil attacks, Newcomer attacks[89], Request Drop attacks, Denial of Service (DoS) attacks, Outage attacks, Sinkhole attacks, Eavesdropping attacks, Composition/Community Exclusion attacks, and Component/Member Exclusion attacks[90]. It is possible for trust evaluation approaches to be vulnerable if they are not adequately protected.

Lack of a comprehensive trust evaluation in the software supply network: The trust evaluation in SECO requires understanding software's dynamic nature as it crosses various hubs across the SECOs. Such SECOs are nurtured and maintained by various actors, encompassing software engineers, package maintainers, and communities, in various hubs, for instance, app stores, repositories, and software package managers[54]. Therefore, in trust evaluation, it becomes pivotal to consider the involvement of hubs and actors within or across SECO(s). This means that we need techniques that track and trace all human and non-human

entities that touch a software artifact before it is run in situ. Furthermore, it introduces two perspectives: (1) SECO centric, where only the local ecosystem is concerned, and (2) SECOs centric, where multiple ecosystems play a part in the creation, delivery, and deployment of a software artifact.

Introduction of immature standards and tools: The utilization of Software Bills of Materials (SBOMs) and Software Composition Analysis (SCA) tools serve as a valuable resource for trust information consumers. By using an SBOM-based supply chain metadata approach, it is possible to track the composition and provenance of every component of a software product, verify the integrity of each component as well as its pedigree, comprehend the intricate components of software products, and utilize this metadata to assess and manage software product risks systematically[91], offering insights into which specific elements may have an impact on SECO trust. We view these developments as crucial building blocks in the trust chain that is built between software engineers and end-users. It is now in the hands of the key players in the software supply chain to use these technologies to provide the right (trust) information in the right amounts at the right time.

Lack of criteria for measuring the accuracy of the trust evaluation: Although trust evaluation approaches vary in accuracy depending on the specific context, objectives, trust factors, and techniques, we currently do not have a standard or benchmark for evaluating their accuracy[58]. As different trust information, consumers interpret trust differently, it is difficult to consistently assess and compare results. Trust evaluations are often used to inform the decision-making process. Without accuracy criteria, decision-makers may make inappropriate choices based on unreliable or unvalidated results, leading to bias and potential risks.

Due to the many complex dimensions around trust in SECOs, it is impossible to develop a comprehensive software trust framework that suffices in all cases, for all software ecosystems, and for all end-user trust requirements. However, we believe that all these trust frameworks should be built on a generic understanding of trust in SECOs as outlined in this and previous work[54]. Furthermore, we expect that new solutions will sprout around ecosystem hubs, such as GitHub and Libraries.io in the near future. Finally, we believe that trust hubs should be engineered, maintained, and sustained by the community of software enthusiasts that has kept so many open-source SECOs going for decades.

5.2 | Our Future Plan

Based on the results of this SM, we intend to develop a platform called TrustSECO, which is dedicated to computing trust scores for various software component versions. The trust facts will be drawn from diverse sources, including websites, online communities, virus scanners, and input from software stakeholders. As a result of these trust facts, we will develop a method for calculating trust scores. The tool will be evaluated with software engineers to determine its effectiveness in facilitating the selection and use of specific software packages[54]. Additionally, we intend to incorporate a number of new technologies into our tool, including (1) Distributed Ledger (DL), which will serve as a repository for these trust scores and associated facts in order to facilitate collaboration among software engineers in order to share information regarding trust. The individuals and organizations using the ledger will have the ability to contribute publicly verifiable facts that can positively or negatively impact the trust scores associated with software packages; (2) Decentralized Autonomous Organization (DAO), which lives on the DL, enables DAO members to coordinate and govern themselves by a set of self-executing rules[92]. It is a sustainable infrastructure that can independently function after the academic project ends; and (3) Natural Language Processing (NLP), enables to extraction of quality attributes expressed in the input text data and determines the sentiment polarity. Moreover, our trust information, including trust scores, will be seamlessly integrated into other platforms, such as software packaging systems. This integration ensures that when software engineers seek to install specific software package versions through these systems, they will receive pertinent trust information before installation.

6 | CONCLUSION

In this article, we study the approaches to evaluating trust in SECO. Our primary objective is to formulate a comprehensive strategy aimed at mitigating the risks posed to the SECO's health stemming from the integration of untrustworthy third-party software components. To achieve this goal, we employed an SM, systematically reviewing and synthesizing insights from 42 selected manuscripts of the state-of-the-art trust measurement within the SECO context. Through a comprehensive examination of the selected manuscripts, this study presents techniques of the evaluation approaches, aspects considered in software trust evaluation, the formats of the evaluation result presentation, as well as the included SECO entities.

Our findings show that there are limited empirical studies conducted on SECO trust. Most of the methods are those that address software trustworthiness and ignore other SECO entities. The majority of these approaches are evidence-based and are generally considered to be multi-criteria decision-making problems [50]. The most commonly used evidence is software security, reliability, maintainability, and safety. The presentation of trust results is diverse depending on the requirements. One-third of the results are in numerical form, i.e., decimals or integers between 0 and 10, and in addition, some are presented in the form of degrees or actions of the decisions.

Based on the findings, we analyze the challenges of the current approaches. Trust depends on the specifics of a given environment and context [22], resulting in different evaluation criteria and attributes. Developing a one-size-fits-all generic approach is therefore a difficult task [63]. However, without such a generic approach, there is a high risk of introducing biases in specific approaches [50]. In addition, software trust is comprehensive, taking into account all actors and hubs of SECO [48]. Therefore, another major challenge lies in expanding the scope of trust evidence or attributes beyond software quality attributes to encompass SECO entities and their corresponding attributes [17]. Future directions should leverage advances in some technologies, for instance, data analytics, blockchain, and deep learning, to explore and analyze diverse data sources, identifying patterns, exceptions, and trends in large data sets to enable objective and accurate trust evaluation.

Systematic Mappings Articles

1. Mohammed Alhamad, Tharam Dillon, and Elizabeth Chang. A trust-evaluation metric for cloud applications. *International Journal of Machine Learning and Computing*, 1(4):416, 2011.

2. Tie Bao, Shufen Liu, and Lu Han. Research on an analysis method for software trustworthiness based on rules. In *The 2010 14th International Conference on Computer Supported Cooperative Work in Design*, pages 43–47. IEEE, 2010.

3. Liang Chen, Ping Cheng, and Wei Liu. The model and method of trustworthiness level evaluation for software product. In *2010 Sixth International Conference on Natural Computation*, volume 2, pages 709–715. IEEE, 2010.

4. Rumei Deng, Yixiang Chen, Hengyang Wu, and Hongwei Tao. Software trustworthiness evaluation using structural equation modeling. *International Journal of Performability Engineering*, 15(10):2628, 2019.

5. Shuai Ding, Xi-Jun Ma, and Shan-Lin Yang. A software trustworthiness evaluation model using objective weight based evidential reasoning approach. *Knowledge and information systems*, 33:171–189, 2012.

6. Shuai Ding, Shan-Lin Yang, and Chao Fu. A novel evidential reasoning based method for software trustworthiness evaluation under the uncertain and unreliable environment. *Expert Systems with Applications*, 39(3):2700–2709, 2012.

7. Shuai Ding and Shanlin Yang. Research on evaluation index system of trusted software. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2008.

8. Wanjiang Han, Suyi Li, Haoran Jia, Tianbo Lu, Jincui Yang, Ruotong Yu, and Yang Li. Research on software trustworthiness evaluation for web application based on software product. *International Journal of u- and e- Service, Science and Technology*, 10:89–104, 01 2017.

9. Hou Hong, Song Qinbao, Yang Jing, and Hao KeGang. The research of bpm software trustworthy evaluation model. In *2009 First International Workshop on Education Technology and Computer Science*, volume 3, pages 815–823. IEEE, 2009.

10. Anne Immonen and Marko Palviainen. Trustworthiness evaluation and testing of open source components. In *Seventh International Conference on Quality Software (QSIC 2007)*, pages 316–321. IEEE, 2007.

11. Rong Jiang. A trustworthiness evaluation method for software architectures based on the principle of maximum entropy (pome) and the grey decision-making method (gdmm). *Entropy*, 16(9):4818–4838, 2014.

12. Noura Limam and Raouf Boutaba. Assessing software service quality and trustworthiness at selection time. *IEEE transactions on software engineering*, 36(4):559–574, 2010.

13. Yu Liu, Shmuel Tyszberowicz, Zhiming Liu, and Bo Liu. itrusteval: A framework for software trustworthiness evaluation with an intelligent ahp-based method. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1778–1785. IEEE, 2022.

14. Gang Lu, Huaimin Wang, and Xiaoguang Mao. Using electre tri outranking method to evaluate trustworthy software. In *Autonomic and Trusted Computing: 7th International Conference, ATC 2010, Xi'an, China, October 26-29, 2010. Proceedings 7*, pages 219–227. Springer, 2010.

15. Li Meng, Zhou Xianzhong, Wang Jiacun, Zhao Jiabao, and Zhu Yingying. A perspective of software trustworthiness based on distrustable factors. In *2009 International Conference on Networking, Sensing and Control*, pages 873–878. IEEE, 2009.

16. Mitra Nami and Witold Suryn. Case study: using requirements and finite state machine for evaluating software trustworthiness. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 3095–3100. IEEE, 2012.

17. Mitra Nami and Witold Suryn. Software trustworthiness: past, present and future. In *Trustworthy Computing and Services: International Conference, ISCTCS 2012, Beijing, China, May 28–June 2, 2012, Revised Selected Papers*, pages 1–12. Springer, 2013.

18. Li Shi and Shanlin Yang. The evaluation of software trustworthiness with fahp and ftopsis methods. In *2009 International Conference on Computational Intelligence and Software Engineering*, pages 1–5. IEEE, 2009.

19. Li Shi, Shanlin Yang, Kai Li, and Ben-gong Yu. Developing an evaluation approach for software trustworthiness using combination weights and topsis. *J. Softw.*, 7(3):532–543, 2012.

20. Hao Song, Dengsheng Wu, Minglu Li, Chen Cai, and Jianping Li. An entropy based approach for software risk assessment: A perspective of trustworthiness enhancement. In *The 2nd International Conference on Software Engineering and Data Mining*, pages 575–578. IEEE, 2010.

21. Yu Song and Ling Wang. Software trusted comprehensive evaluation model based on fuzzy grey method. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 1, pages 513–516. IEEE, 2010.

22. George Spanoudakis. Dynamic trust assessment of software services. In *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting*, pages 36–40, 2007.

23. Hongwei Tao. Software trustworthiness evaluation method based on relationships between criteria. In *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, pages 403–408. IEEE, 2022.

24. Hongwei Tao, Qiaoling Cao, Haoran Chen, Yang Xian, Songtao Shang, and Xiaoxu Niu. A novel software trustworthiness evaluation strategy via relationships between criteria. *Symmetry*, 14(11):2458, 2022.

25. Hongwei Tao, Yixiang Chen, Han Liu, Hengyang Wu, and Yinghui Hu. Attributes oriented software trustworthiness measure based on axiomatic approaches. *Journal of Internet Technology*, 23(3):583–592, 2022.

26. Hongwei Tao, Yixiang Chen, and Hengyang Wu. A reallocation approach for software trustworthiness based on trustworthy attributes. *Mathematics*, 8(1):14, 2019.

27. Hongwei Tao, Yixiang Chen, and Hengyang Wu. Theoretical and empirical validation of software trustworthiness measure based on the decomposition of attributes. *Connection Science*, 34(1):1181–1200, 2022.

28. Hongwei Tao, Hengyang Wu, and Yixiang Chen. An approach of trustworthy measurement allocation based on sub-attributes of software. *Mathematics*, 7(3):237, 2019.

29. Hongwei Tao and Jie Zhao. Research of software trustworthiness measurement based on validation. In *2016 International Symposium on System and Software Reliability (ISSSR)*, pages 7–12. IEEE, 2016.

30. Hongwei Tao and Jie Zhao. Source codes oriented software trustworthiness measure based on validation. *Mathematical Problems in Engineering*, 2018:1–10, 2018.

31. Junfeng Tian and Yuhui Guo. Software trustworthiness evaluation model based on a behaviour trajectory matrix. *Information and Software Technology*, 119:106233, 2020.

32. Ben Wang, Xingshe Zhou, Gang Yang, and Yalei Yang. Ds theory-based software trustworthiness classification assessment. In *2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing*, pages 434–438. IEEE, 2010.

33. Yonghao Wang, Guangping Zeng, Qi Wang, Qingchuan Zhang, et al. Research on trustworthiness evaluation method of software resources based on fuzzy sets. *J. Softw.*, 8(12):3214–3221, 2013.

34. Zheng Yan. A comprehensive trust model for component software. In *Proceedings of the 4th international workshop on Security, privacy and trust in pervasive and ubiquitous computing*, pages 1–6, 2008.

35. Zheng Yan and Christian Prehofer. Autonomic trust management for a component-based software system. *IEEE Transactions on Dependable and Secure Computing*, 8(6):810–823, 2010.

36. Jian Yang. A classification evaluation model for software trustworthiness based on trustworthiness evolution. In *2011 International Conference on Business Management and Electronic Information*, volume 1, pages 222–227. IEEE, 2011.

37. Xi Yang, Gul Jabeen, Ping Luo, Xiao-Ling Zhu, and Mei-Hua Liu. A unified measurement solution of software trustworthiness based on social-to-software framework. *Journal of Computer Science and Technology*, 33:603–620, 2018.

38. Ye Yang, Qing Wang, and Mingshu Li. Process trustworthiness as a capability indicator for measuring and improving software trustworthiness. In *Trustworthy Software Development Processes: International Conference on Software Process, ICSP 2009 Vancouver, Canada, May 16-17, 2009 Proceedings*, pages 389–401. Springer, 2009.

39. Lewen Zhang, Yong Zhou, Yixiang Chen, Min Zhang, and Juyang Zhang. Stability of software trustworthiness measurements models. In *2013 IEEE Seventh International Conference on Software Security and Reliability Companion*, pages 219–224. IEEE, 2013.

40. Xiao Zhang, Wei Li, ZhiMing Zheng, and BingHui Guo. Optimized statistical analysis of software trustworthiness attributes. *Science China Information Sciences*, 55:2508–2520, 2012.

41. Yuejin Zhang, Yanmei Zhang, and Mo Hai. An evaluation model of software trustworthiness based on fuzzy comprehensive evaluation method. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 616–619. IEEE, 2012.

42. ZhiMing Zheng, ShiLong Ma, Wei Li, Xin Jiang, Wei Wei, LiLi Ma, and ShaoTing Tang. Complexity of software trustworthiness and its dynamical statistical analysis methods. *Science in China Series F: Information Sciences*, 52(9):1651–1657, 2009.

References

43. Jansen S, Cusumano MA, Brinkkemper S. *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar, 2013.

44. Manikas K, Hansen KM. Software ecosystems–A systematic literature review. *Journal of Systems and Software.* 2013;86(5):1294–1306.

45. Sonatype . 2022 state of the software supply chain. In: *The 8th Annual Report on Global Open Source Software Development*Sonatype Inc. 2022.

46. Rosemann M. Trust-aware process design. In: *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*Springer. 2019:305–321.

47. Hou F, Jansen F, Vries dA, Jansen S. The Role of Software Trust in Selection of Open-Source and Closed Software. In: *2023 IEEE/ACM 11th International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS)*IEEE. 2023:30–37.

48. Hou F, Jansen S. A systematic literature review on trust in the software ecosystem. *Empirical Software Engineering.* 2023;28(1):8.

49. Lewis JD, Weigert A. Trust as a social reality. *Social forces.* 1985;63(4):967–985.

50. Pal S, Hill A, Rabehaja T, Hitchens M. A blockchain-based trust management framework with verifiable interactions. *Computer Networks.* 2021;200:108506.

51. Hillebrand C, Coetzee M. Towards Reputation-as-a-Service. In: *2013 Information Security for South Africa*IEEE. 2013:1–8.

52. Cho JH, Xu S, Hurley PM, Mackay M, Benjamin T, Beaumont M. Stram: Measuring the trustworthiness of computer-based systems. *ACM Computing Surveys (CSUR).* 2019;51(6):1–47.

53. Ray I, Chakraborty S. A vector model of trust for developing trustworthy systems. In: *Computer Security–ESORICS 2004: 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, September 13-15, 2004. Proceedings 9*Springer. 2004:260–275.

54. Hou F, Farshidi S, Jansen S. TrustSECO: A Distributed Infrastructure for Providing Trust in the Software Ecosystem. In: *International Conference on Advanced Information Systems Engineering*Springer. 2021:121–133.

55. Guo G, Zhang J, Thalmann D, Basu A, Yorke-Smith N. From ratings to trust: an empirical study of implicit trust in recommender systems. In: *Proceedings of the 29th annual acm symposium on applied computing* 2014:248–253.

56. Cho JH, Chan K, Adali S. A survey on trust modeling. *ACM Computing Surveys (CSUR).* 2015;48(2):1–40.

57. Sabater J, Sierra C. Review on computational trust and reputation models. *Artificial intelligence review.* 2005;24:33–60.

58. Li X, Moreschini S, Zhang Z, Taibi D. Exploring factors and metrics to select open source software components for integration: An empirical study. *Journal of Systems and Software.* 2022;188:111255.

59. Group TC. Trusted Platform Model (TPM) 2.0: A Brief Introduction.; 2019.

60. Zheng Z, Ma S, Li W, et al. Dynamical characteristics of software trustworthiness and their evolutionary complexity. *Science in China Series F: Information Sciences.* 2009;52(8):1328–1334.

61. Alhanahnah M, Bertok P, Tari Z, Alouneh S. Context-aware multifaceted trust framework for evaluating trustworthiness of cloud providers. *Future Generation Computer Systems.* 2018;79:488–499.

62. Fong E, Kass M, Rhodes T, Boland F. Structured assurance case methodology for assessing software trustworthiness. In: *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement Companion*IEEE. 2010:32–33.

63. Paulus S, Mohammadi NG, Weyer T. Trustworthy software development. In: *IFIP International Conference on Communications and Multimedia Security*Springer. 2013:233–247.

64. Sherchan W, Nepal S, Paris C. A survey of trust in social networks. *ACM Computing Surveys (CSUR).* 2013;45(4):1–33.

65. Jøsang A, Ismail R, Boyd C. A survey of trust and reputation systems for online service provision. *Decision support systems.* 2007;43(2):618–644.

66. Su Z, Liu L, Li M, Fan X, Zhou Y. ServiceTrust: Trust management in service provision networks. In: *2013 IEEE international conference on services computing*IEEE. 2013:272–279.

67. Houser D, Wooders J. Reputation in auctions: Theory, and evidence from eBay. *Journal of Economics & Management Strategy.* 2006;15(2):353–369.

68. Huynh TD, Jennings NR, Shadbolt NR. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems.* 2006;13:119–154.

69. Artz D, Gil Y. A survey of trust in computer science and the semantic web. *Journal of Web Semantics.* 2007;5(2):58–71.

70. Olmedilla D, Rana OF, Matthews B, Nejdl W. Security and trust issues in semantic grids. In: *Dagstuhl Seminar Proceedings*Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2006.

71. Costante E, Den Hartog J, Petković M. Understanding perceived trust to reduce regret. *Computational Intelligence.* 2015;31(2):327–347.

72. Lu J, Wu D, Mao M, Wang W, Zhang G. Recommender system application developments: a survey. *Decision Support Systems.* 2015;74:12–32.

73. Lu J, Shambour Q, Xu Y, Lin Q, Zhang G. BizSeeker: a hybrid semantic recommendation system for personalized government-to-business e-services. *Internet Research.* 2010.

74. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering.* 2005;17(6):734–749.

75. Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th international conference on World Wide Web* 2001:285–295.

76. Fu C, Yang SL. The group consensus based evidential reasoning approach for multiple attributive group decision analysis. *European Journal of Operational Research.* 2010;206(3):601–608.

77. Zhang Q, Yu T, Irwin K. A Classification Scheme for Trust Functions in Reputation-Based Trust Management.. In: *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*. 127. Citeseer. 2004.

78. Su Q, Huang CR, Chen HK. Evidentiality for text trustworthiness detection. In: *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground* 2010:10–17.

79. Mujahid S, Abdalkareem R, Shihab E. What are the characteristics of highly-selected packages? A case study on the npm ecosystem. *arXiv preprint arXiv:2204.04562.* 2022.

80. Hu Y, Zhang J, Bai X, Yu S, Yang Z. Influence analysis of Github repositories. *SpringerPlus.* 2016;5(1):1–19.

81. Saini M, Verma R, Singh A, Chahal KK. Investigating diversity and impact of the popularity metrics for ranking software packages. *Journal of Software: Evolution and Process.* 2020;32(9):e2265.

82. Paul J. The TIOBE Quality Indicator, a pragmatic way of measuring code quality. In: TIOBE. TIOBE 2023.

83. Petersen K, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and software technology.* 2015;64:1–18.

84. Carrera-Rivera A, Ochoa W, Larrinaga F, Lasa G. How-to conduct a systematic literature review: A quick guide for computer science research. *MethodsX.* 2022;9:101895.

85. Jadhav AS, Sonar RM. Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach. *Journal of Systems and Software.* 2011;84(8):1394–1407.

86. Vulpen vP, Jansen S, Brinkkemper S. The orchestrator's partner management framework for software ecosystems. *Science of Computer Programming.* 2022;213:102722.

87. Wang J, Jing X, Yan Z, Fu Y, Pedrycz W, Yang LT. A survey on trust evaluation based on machine learning. *ACM Computing Surveys (CSUR).* 2020;53(5):1–36.

88. Amoroso E, Nguyen T, Weiss J, Watson J, Lapiska P, Starr T. Toward an approach to measuring software trust. In: *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*IEEE Computer Society. 1991:198–198.

89. Sun YL, Han Z, Yu W, Liu KR. Attacks on trust evaluation in distributed networks. In: *2006 40th Annual Conference on Information Sciences and Systems*IEEE. 2006:1461–1466.

90. Wahab OA, Bentahar J, Otrok H, Mourad A. A survey on trust and reputation models for Web services: Single, composite, and communities. *Decision Support Systems.* 2015;74:121–134.

91. Clancy C, Ferraro J, Martin R, Pennington A, Sledjeski C, Wiener C. Deliver uncompromised: Securing critical software supply chains. *MITRE Technical Papers.* 2021;24:01.

92. Vulpen vP, Jansen S. Decentralized Autonomous Organization Design for the Commons and the Common Good. *Available at SSRN 4418782.* 2023.