Archives of Computational Methods in Engineering (2023) 30:2041–2080
https://doi.org/10.1007/s11831-022-09864-y

REVIEW ARTICLE

Software Development Analytics in Practice: A Systematic Literature Review

João Caldeira[1] · Fernando Brito e Abreu[1] · Jorge Cardoso[2,3] · Rachel Simões[4] · Toacy Oliveira[4] · José Pereira dos Reis[5]

Received: 26 March 2022 / Accepted: 19 November 2022 / Published online: 10 January 2023

Abstract

Software development analytics is a research area concerned with providing insights to improve product deliveries and processes. Many types of studies, data sources and mining methods have been used for that purpose. This systematic literature review aims at providing an aggregate view of the relevant studies on Software Development Analytics in the past decade, with an emphasis on its application in practical settings. Definition and execution of a search string upon several digital libraries, followed by a quality assessment criteria to identify the most relevant papers. On those, we extracted a set of characteristics (study type, data source, study perspective, development life-cycle activities covered, stakeholders, mining methods, and analytics scope) and classified their impact against a taxonomy. Source code repositories, exploratory case studies, and developers are the most common data sources, study types, and stakeholders, respectively. Testers also get moderate attention from researchers. Product managers' concerns are being addressed frequently and project managers are also present but with less prevalence. Mining methods are rapidly evolving, as reflected in their identified long list. Descriptive statistics are the most usual method followed by correlation analysis. Being software development an important process in every organization, it was unexpected to find that process mining was present in only one study. Most contributions to the software development life cycle were given in the quality dimension. Time management and costs control were less prevalent. The analysis of security aspects is even more reduced, however, evidences suggest it is an increasing topic of concern. Risk management contributions are also scarce. There is a wide improvement margin for software development analytics in practice. For instance, mining and analyzing the activities performed by software developers in their actual workbench, i.e., in their IDEs. Together with mining developers' behaviors, based on the evidences and trend, in a short term period we expect an increase in the volume of studies related with security and risks management.

✉ João Caldeira
jcppc@iscte-iul.pt

Fernando Brito e Abreu
fba@iscte-iul.pt

Jorge Cardoso
jcardoso@dei.uc.pt

Rachel Simões
rachelvital@cos.ufrj.br

Toacy Oliveira
toacy@cos.ufrj.br

José Pereira dos Reis
jvprs@iscte-iul.pt

[1] Iscte - Instituto Universitário de Lisboa, ISTAR-Iscte, Lisboa, Portugal

[2] University of Coimbra, Coimbra, Portugal

[3] Huawei Munich Research Center, Munich, Germany

[4] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

[5] Iscte - Instituto Universitário de Lisboa, ISTAR-Iscte, Lisbon, Portugal

1 Introduction

Defining new processes and allocating the right resources, particularly for large organizations, is a challenging task for software project managers, primarily because it requires acquaintance on existing processes and tools, the understanding of different stakeholders, and the coordination of technical expertise in multiple domains [38]. Failing to properly manage these various aspects, namely when decisions are based on "gut feeling" (often dubbed "personal experience from past projects") may cause software development projects to produce hard to maintain technical artifacts, to surpass budget and schedule, and deliver defective products [19, 43].

The Software Development Analytics (SDA) research field aims at mitigating the aforementioned risks by providing the stakeholders' decision-making process with structured data-driven pieces of evidence, such as insights on software products and processes.

1.1 Motivation

The term "software analytics" (SA) emerged naturally expressing the work of several research groups aiming to expand the traditional scope on analyzing software artifacts by means of mining software repositories [77]. These groups conducted cutting-edge research and technology innovation in an interdisciplinary area that spans across big data, machine learning, systems, and software engineering. This approach led software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for completing various tasks around software systems, software users, and software development processes [76].

Software Development Analytics (SDA), the adoption of analytics methods with the focus on the management of software development projects, was proposed in [6]. It differs from software cybernetics, which is a subdivision of "cybernetics" in the domain of software engineering [73]. It references the description of "cybernetics" by Wiener, if software is regarded as part of the machine, and can be defined simply as communication and control in software. However, most researchers in the area believe software cybernetics is more diverse in scope. In fact, it is described as the interplay between software or software behaviour and control [8]. In its simplest form, the field of software cybernetics treated software problems and control problems in an integrated way [9].

In turn, SDA is broader in its scope and based on a structured framework to identify adequate resources, ask meaningful questions, collect and analyze information properly, provide insights to the stakeholders and finally to identify the benefits for the software development lifecycle, either by looking at its past, present or future perspectives [6]. Although a few aspects of software cybernetics may seem to overlap with SDA, for instance, the software development activities, the mining method or the type of study, many others are missing, such as the stakeholders, the analytics scope and more importantly, the potential contributions towards the relevant properties of software projects and where those can effectively support the decisions taken by managers.

Since the time analytics was proposed for the practice of developing software, a vast amount of literature was produced presenting stakeholders with new ways of improving the efficiency and effectiveness in developing software products, by providing insights on how to streamline the processes or to optimize resource allocation [1].

1.2 Contributions

A decade has elapsed since the first discussions on methodologies, techniques and tools to boost the adoption of analytics in the software development practice. However, there have been a small number of reports on the practice impact or benefits that software development analytics results have created on software development [77]. This systematic literature review (SLR) identifies, analyzes and aggregates the relevant primary studies in this period, following a well defined protocol, aligned with the best practices [18, 32]. Its main objectives are to:

- summarize the main types of empirical studies performed, target software life cycle activities, and corresponding data sources;
- identify the mining methods and analytics that were applied;
- evaluate the contributions of the selected primary studies;
- define a taxonomy to classify the impact provided by each primary study on software development dimensions such as: quality/technical debt, time, costs, risks and security.

This paper is organized as follows: Sect. 2 provides background related to the research area and emphasizes the differences between this and previous systematic reviews in the domain. We outline the research methodology and systematic review planning in Sect. 3, present the systematic review execution, data analysis, results discussion and threats to validity in Sect. 4, and the concluding comments appear in Sect. 5.

2 Background

Mining software repositories is currently a widespread method to gather insights from the software development process [23, 40, 48]. As these methods evolved, the software engineering practice took advantage of lessons learned and applied them in real live scenarios [39]. The last decade has seen the birth of a multitude of analytics related companies, solutions and methodologies [39, 48, 63], often powered by machine learning techniques. It was also a period where process mining saw boundless adoption in several business domains [22, 64, 65]. Both approaches, machine learning and process mining, are nowadays being used to reduce the costs of producing software products, to improve their quality, reduce time-to-market, and support the decision making-process.

2.1 Related Work

Many SLRs have been published in the field of software engineering [32]. However, the ones addressing SDA concerns, from a holistic perspective, are scarce and often insufficiently detailed, since several aspects we deem relevant to advance the current state of the art are lacking or did not have exhaustive scrutiny. Notwithstanding, we briefly describe hereinafter all the systematic reviews whose scope somehow intersects the usual topics of SDA.

A SLR covering primary studies from 2000 to 2014, aiming to identify gaps in knowledge and open research areas in SA was presented in [1]. It considered 19 primary studies out of 135 and the authors concluded that the practitioners who benefited most from SA studies were developers, testers, project managers (PM), portfolio managers, and higher management, with 47% of the considered studies supporting only developers. Maintainability and reverse engineering, team collaboration and dashboards, incident management and defect prediction, the SA platform, and software effort estimation were among the domains mostly studied, with 47% of them analyzing only one artifact. Based on their analysis, since most of the research addresses only the low-level analytics of source code, the authors recommended researchers to use more datasets, to achieve higher confidence level in the results. They also suggested to target higher-level business decision making profiles, like portfolio management, marketing strategy, and sales directions.

A survey of the publicly available repositories and the classification of the most common ones is presented in [54]. Authors also discussed the problems faced by researchers when applying machine learning or statistical techniques to them. The conclusions highlight the fact that some of the problems, such as outliers or noise, have been extensively studied in software engineering, whilst others need further research. They authors pointed out the need of further research work to deal with the imbalance and data shifting from the machine learning point of view and replication of primary studies.

A mapping study on the investigation of frequently applied empirical methods, targeted research purposes, used data sources, and applied data processing approaches and tools in empirical software engineering (ESE) was reported in [78]. The goal was to identify new trends and obtain interesting observations of ESE across different sub-fields of software engineering on 538 selected articles from January 2013 to November 2017. The authors observed that the trend of applying empirical methods in software engineering is continuously increasing and the most commonly applied methods are experiments, case studies and surveys, with open source projects being frequently used as data sources.

A systematic mapping study aiming at identifying the quantity, topic, and empirical methods used, targeting the analysis of how software development practices are influenced by the use of a distributed social coding platform like GitHub, was presented in [13]. The authors assessed 80 publications from 2009 to 2016, and the results showed that most works focus on the interaction around coding-related tasks and project communities. They also identified some concerns about how reliable were those results based on the fact that, overall, papers used small data sets and poor sampling techniques, employed a scarce variety of methodologies and/or were hard to replicate. As a conclusion, they attested the high activity of research work around the field of open source collaboration, identified shortcomings and proposed actions to mitigate them.

A systematic mapping study providing an overview of the concerns addressed in the different phases of the software development life cycle (SDLC), was published in [15]. Results are reported from different viewpoints and conclusions highlight that there is a considerable variation in the use of terminologies and addressing concerns in different phases of the SDLC.

Inspired by the increasing usage of data analytics in all areas of science and engineering, a systematic mapping study, aiming to investigate the usage of different types of analytics for software project management was presented in [47]. The authors analyzed the accessibility of the data, as well as the degree of validation reported in the final 115 studies selected for appraisal. Results provided evidences that the majority of studies were focusing on predictive and prescriptive analytics, with almost half of the studies being essentially predictive. When comparing information versus insight as the direction of analytics, the authors found that information oriented analytics (descriptive and predictive) had a greater number of

related studies (60% of papers) than analytics searching for insight (diagnostic or prescriptive). As a final remark, their systematic mapping findings was compared with the results obtained by [7].

A systematic mapping study published in [3] aims at providing an overview of the sub-domains, contribution types, research types, research methods and identify the role of software analytics in the field of "green software engineering". Findings show, that 163 papers out of the 260 initially found on digital libraries, used software analytical methods like statistical analysis and static analysis. Furthermore, only 11 out of the 50 papers kept for final data extraction, used software analytics techniques to foster green software engineering. Results revealed the need to develop new/improved automated software analytics tools for software practitioners, along with metrics explaining the correlation between energy usage and other quality attributes.

Our SLR aims to expand the existing knowledge about SDA, by adapting and extending the data perspectives, dimensions, and concerns identified and used by the above works. The target properties we deem as most important for a primary study to be considered relevant in this SLR are the following:

- **Quality.** To assess the delivery of a good product or project outcome.
- **Scope.** To evaluate the meeting of requirements and objectives.
- **Time.** To track the project delivering on time.
- **Cost.** To manage the delivery within estimated cost and effort.
- **Reusability.** The use of existing assets in some form within the software product development process.
- **Maintainability.** To asses the degree to which an application is understood, repaired, or enhanced.
- **Evolvability.** Used to describe a multifaceted quality attribute to evaluate a software system's ability to easily accommodate future changes.
- **Performance.** To measure how effective a software system is with respect to the allocation of resources and correspondent time constraints.
- **Security.** A cross-cutting appraise that takes into account mechanisms, such as access control, and robust design to prevent software attacks.
- **Risk.** To address the possibility that one or more of the above properties are exposed to such levels of uncertainty that may lead them to produce undesired outcomes.

Based on this set, we propose a taxonomy to classify primary studies.

3 Research Methodology

In contrast to a non-structured review process, a SLR reduces bias and follows a precise and rigorous sequence of methodological steps to research literature [31, 68]. A SLR relies on a well-defined and evaluated review protocols to search, extract, analyze, and document results as stages. This section describes the methodology applied for those activities.

3.1 Planning the Review

3.1.1 Research Questions

This SLR is driven by the following research questions:

RQ1. *What type of empirical studies have been conducted in SDA?*

Justification. The list of the main types of studies reported in SDA literature can provide a comprehensive view, both for practitioners and researchers, not only to identify areas of opportunity, but also to optimize established methods.

RQ2. *What are the main data sources used for SDA related studies?*

Justification. Identifying those data sources is helpful, to provide soundness to the corresponding studies, to facilitate replication, and to stimulate the appearance of new datasets to address knowledge gaps in the field.

RQ3. *What type of process/project perspective analysis was conducted?*

Justification. It refers to the ability to identify if the studies are being done before **(pre-mortem)** or after **(post-mortem)** a process/project is finished. While the latter is more frequent, namely due to the use of existing software repositories, a pre-mortem perspective can add additional value in the decision making process, as taking corrective actions on a timely manner is fundamental to keep projects or processes on track.

RQ4. *What are the most studied SDLC activities?*

Justification. Understanding what SDLC activities are targeted the most (and those that are not), will help practitioners identify where most concerns and challenges are within the software development practice. It can also contribute to open new research streams to foster a deeper understanding of the complete SDLC.

RQ5. *Who were the target stakeholders of these studies?*

Justification. Software projects are risky to conduct and continue to be difficult to predict [6]. SDA in practice, holds out the promise to provide decision-makers with data-driven evidences in order to better manage risk, improve efficiency and effectiveness on development projects. Studies should address the needs of different stakeholders. Identifying those beneficiaries is vital to understand if the right tools, methods and insights are reaching the ones that most need support on their daily activities.

RQ6. *What are the main mining methods being used?*

Justification. Assessing the types of mining methods utilized helps to comprehend deeper the goals of past and current research, the limitations of their methods, benefits and conclusions and, highlight opportunities for novel approaches in future research.

RQ7. *Which type/form of analytics was applied?*

Justification. When exploring large volumes of data and many types of metrics, one may exploit different levels of analytics; **descriptive/diagnostics, predictive and prescriptive** [16]. Providing stakeholders in the development process with deep insights and potentially prescribing actions to take under certain circumstances is desirable. Predicting the future and prescribing actions are advanced forms of analytics which researchers and practitioners in the software development domain are expected to use.

RQ8. *What were the relevant contributions to the SDLC ?*

Justification. On every single software development study, we should have clear benefits identified, either from using a new tool or by improving a process using a specific method. Failing to do so, reduces substantially the interest we may find in that literature and shortens the applicability of those methods in the field. SDA in practice is expected to contribute at least (but not limited to) to the following areas of concern in a software project: **technical debt/quality, costs, time, risk and security**.

3.1.2 Search Strategy

Search Terms. Based on the research questions, keywords were extracted and used to search the primary study sources. The search string included the main terms from the topics being researched, including synonyms, related items and alternative spelling. It is based on the same strategy used by [11] and is presented as follows:

("software analytics" OR "software development analytics") AND ("process mining" OR "data mining" OR "big data" OR "data science") AND ("study" OR "empirical" OR "evidence based" OR "experimental" OR "in vivo")

Digital Libraries Searched. A significant phase in a SLR is the search for relevant literature within the domain under study. To search for all the available literature pertinent to our research questions, in addition to some articles we added manually, the following digital libraries were queried:

- ACM Digital Library
- IEEE Xplore
- ScienceDirect
- Scopus
- SpringerLink
- Web of Science
- Wiley Online
- Google Scholar

Publications Time Frame. As mentioned earlier, the SDA research field emerged approximately a decade ago. Since then, as studies have gained a more structured and formal approach, it makes sense to only account for publications in journals, conferences papers, workshops and book chapters, starting from January, 1st of 2010 till the end of 2021.

3.1.3 Selection Criteria

We selected the above libraries based on the eagerness of collecting as many articles/papers as possible, not only because they are recognized as the most representative for Software Engineering research. Google Scholar was selected to account for articles eventually not yet published, but also relevant to the software development domain.

Search Stages Overview The outputs of the process followed to conduct the search is depicted in Fig. 8 in Appendix 1. It compounds 4 sequential stages, which are described as:

Stage 1—Retrieve automatically results from the digital libraries The referred libraries were searched using the specific syntax of each database. The search was configured in each repository to select only papers carried out within the prescribed period. The automatic search was later complemented by a manual search, according to the guidelines of Wohlin [68].

Stage 2—Read titles and abstracts to identify potentially relevant studies Identification of potentially relevant studies based on the analysis of title and abstract. Studies that are clearly irrelevant to the search and duplicates were discarded across the digital libraries. If there was any doubt about

Table 1 Exclusion and inclusion criteria applied at Stage 3

Criterion	Description
Exclusion criteria (EC)	
EC1	Studies published before 2010
EC2	Studies not written in English
EC3	Studies not related to the software development process
EC4	Studies not supported by data collected on any well designed experiment or did not use empirical data from a third party
EC5	Studies merely theoretical or based on expert opinion without locating a specific experience, such as: editorials, prefaces, summaries of articles, interviews, news, analysis/reviews, readers' letters, summaries of tutorials, workshops, panels, round tables, keynotes and poster sessions
EC6	Studies aiming only at describing new development tools or works with the goal of simply assessing and/or validating new analytical methods without a clear statement to the benefits they may provide for the SDLC
Inclusion criteria (IC)	
IC1	Publications should be "journal" or "conference" or "workshop" or "book"
IC2	Works that put validated analytical methods into practice with the goal of understanding and/or improving the software development process
IC3	Articles that clearly addressed any of the analytics depth (RQ7) and provided benefits for the SDLC on any dimension identified in RQ8

Table 2 Quality criteria

Criterion	Description
QC1	Is the paper based on research (or merely a "lessons learned" report based on expert opinion)?
QC2	Is there a clear statement of the aims of the research?
QC3	Is there an adequate description of the context in which the research was carried out?
QC4	Was the research design appropriate to address the aims of the research?
QC5	Was the recruitment strategy appropriate to the aims of the research?
QC6	Was there a control group with which to compare treatments?
QC7	Was the data collected in a way that addressed the research issue?
QC8	Was the data analysis sufficiently rigorous?
QC9	Has the relationship between researcher and participants been adequately considered?
QC10	Are the datasets available to the public, thus allowing replication ?
QC11	Is there a clear statement of findings?
QC12	Is the study of value for research or practice?
QC13	Did the study identified any clear benefits for the SDLC according to RQ8?

whether a study should be included or not, it was included for consideration in a later stage.

Stage 3—Apply inclusion and exclusion criteria on reading the introduction, methods and conclusion Selected studies in previous stages were reviewed, by reading the introduction, methodology section and conclusion. Afterwards, exclusion and inclusion criteria were applied as defined in Table 1. At this stage, in case of doubt preventing a conclusion, the study was read in its entirety.

Stage 4—Obtain primary studies and assess them - A list of primary studies was obtained and later submitted to critical examination using the 13 quality assessment criteria which is set out in Table 2.

3.1.4 Quality Assessment

The strategy to evaluate the quality of the studies is based on a checklist with thirteen criteria. The criteria were based on good practices for conducting empirical research [32] and in the Critical Appraisal Skills Programme (CASP) used in different types of publications [18].

The criteria developed to assess quality covered four main quality issues considered necessary when evaluating primary papers:

- **Reporting.** Three criteria (QC1-QC3) assess if the rationale, goals and context have been clearly stated.
- **Rigor.** Five criteria (QC4-QC8) evaluate if a meticulous and convenient approach have been applied.

- **Credibility.** Two criteria (QC9-QC10) check if the findings are well presented and the gathered insights are plausible and/or credible.
- **Relevance.** The remain criteria (QC11-QC13) are related with the relevancy of the study for the SDLC, stakeholders and the research community.

Selection of primary studies. The quality of each publication should be assessed by the authors after the selection process in Stage 3. The checklist presented in Table 2 was used to assess the credibility and thoroughness of the selected publications. The steps that guided the selection of primary studies to reach the final results, are presented in Fig. 8 in Appendix 1.

Each of the 13 questions was marked as "**Yes**", "**Partially**" or "**No**". We considered a question answered as "**Partially**" in cases where we could derive relevant contents from the text, even if the details were not clearly reported. These answers were scored as follows: "**Yes**"**=1**, "**Partially**"**=0.5**, and "**No**"**=0**. For each selected study, its quality score was computed by summing up the scores of the answers to all the quality criteria questions, being the minimum value admissible "**0**" and the maximum "**13**", in case all the questions were marked with a "**1**".

To provide validation and credibility in the quality assessment, and due to the ordinal scale for the quality criteria score, we computed using a random sample of the 173 articles, the intraclass correlation value between the raters. The results are presented later in Table 4 in Sect. 3.2.2. Whenever agreement was not possible, the first author choice was taken into consideration.

3.1.5 Data Extraction

To gather standard information regarding the papers under analysis, we created a data collection form as represented in Table 10 in Appendix 1. This data collection form helped us to identify the date, venue and authors of the publications and also how each of them addressed the topics of our research questions.

3.1.6 Data Synthesis

The synthesis aimed at grouping findings from the studies in order to: identify the answers to the **RQs** presented earlier in Sect. 3.1 and were organized in a spreadsheet form. This data extraction process was manually conducted by the main author. The spreadsheet was loaded and analyzed using the R statistical engine[1] and has now been disclosed[2].

Obtained results, plots and findings are presented and discussed in Sect. 3.2.

Table 3 Digital library initial search and stages

Digital Library		Stages			
		1	2	3	4
Libraries (ACM, Scopus, Web of Science, Science Direct, IEEE, Wiley Online, SpringerLink)	1144				
	|−− >	3154	681	173	42
Google Scholar and Manually Added	2010				
Total (Input for Stage 1)	3154				

Table 4 Intraclass correlation (ICC) (95% confidence interval)

Subjects	Raters	ICC	Model	Type
30	2	0.801	OneWay	Agreement

3.2 Conducting the Review

This phase is responsible for executing the actions defined in Sect. 3.1.

3.2.1 Execute Search

We started the review with an automatic search followed by a manual search and afterwards applied the inclusion/exclusion criteria. The search as detailed in Sect. 3.1.2, was performed in mid 2019 and updated in the end of the last quarter of 2021, with the search string syntax being adapted to support the different search engines. Initially we identified 3154 articles, and upon reading their titles and abstracts, the dataset was reduced to 681 articles. Following, we filtered them with the inclusion and exclusion criteria. Table 3 and Fig. 8 in Appendix 1, present the summary results and workflow, respectively, for this research.

3.2.2 Apply Quality Assessment Criteria

The selection criteria was based on exclusions and inclusions. Table 1, defined, in Sect. 3.1.3 those criteria used to assess remaining works in *Stage 3*. In case of any doubt, the study was kept for analysis at a later stage. *Stage 3* provided

[1] https://www.r-project.org, https://rstudio.com

[2] doi:10.17632/d3wdzgz88s.2

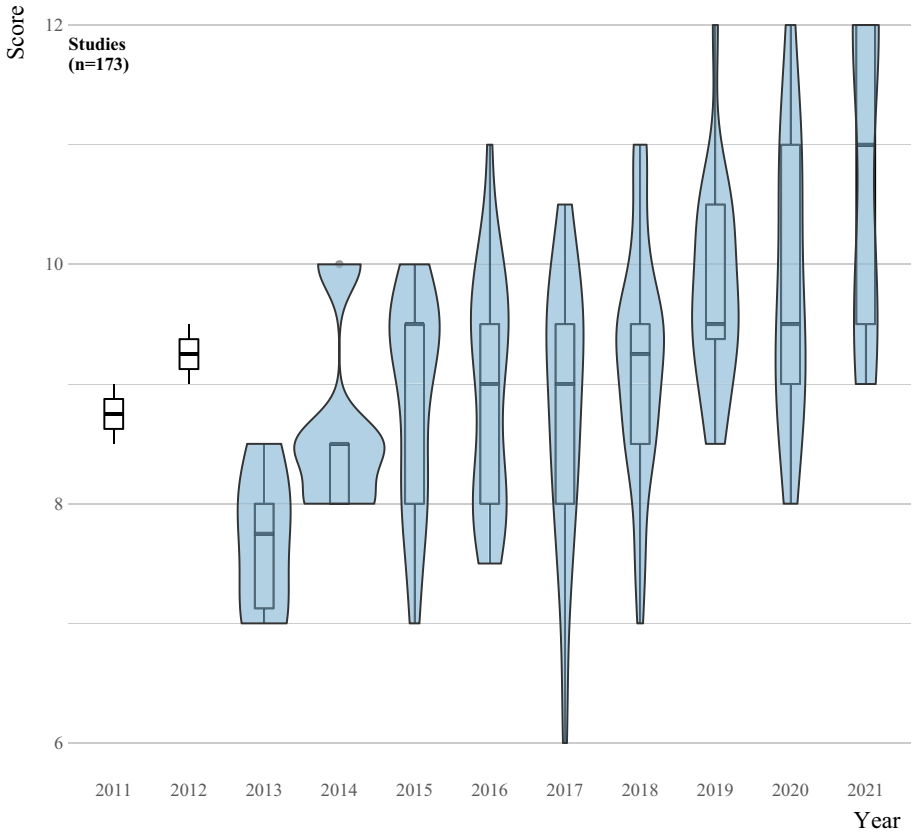

Fig. 1 Studies score per year at Stage 4

as inputs for *Stage 4*, 173 articles, which were then assessed in their quality dimension. At *Stage 4*, we applied the quality criteria described in Sect. 3.1.4, resulting in 42 articles to further extract data and to answer the eight research questions (Table 4).

We classified the studies quality level by plotting their descriptive statistics and analyzing the correspondent quartiles:

– **Min**: 6, **1st Q.**: 8.5, **Median**: 9.0, **Mean**: 9.007, **3rd Q.**: 9.5, **Max**: 12

As seen above, the third quartile is at score **9.5**, therefore, we selected only the studies scoring above that mark. Based on the high level of quality, 42 studies were selected for final data extraction. Figure 1 shows the distribution of all studies per Year right after the quality assessment scoring task.

4 Document the Review

All selected studies and the details to support the statistics we show in Sect. 4.1, are presented in Table 11 in Appendix 2. In Sect. 4.2, we present the main findings, comments and answers to each of the research questions.

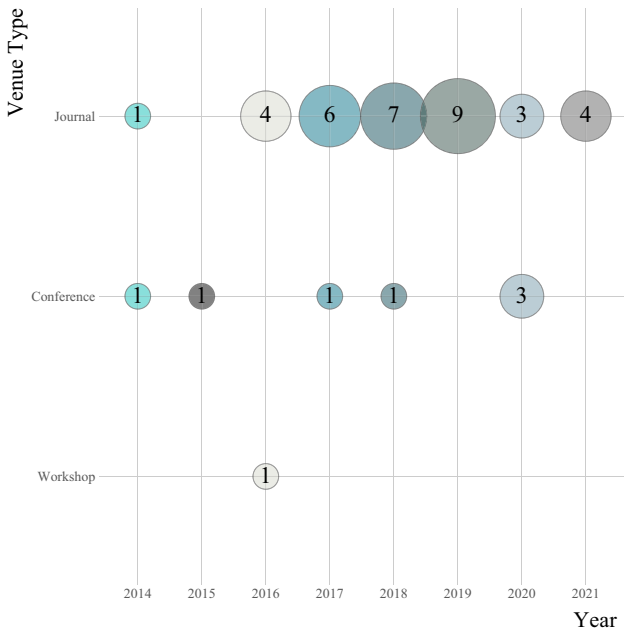

Fig. 2 Number of studies per venue type per year

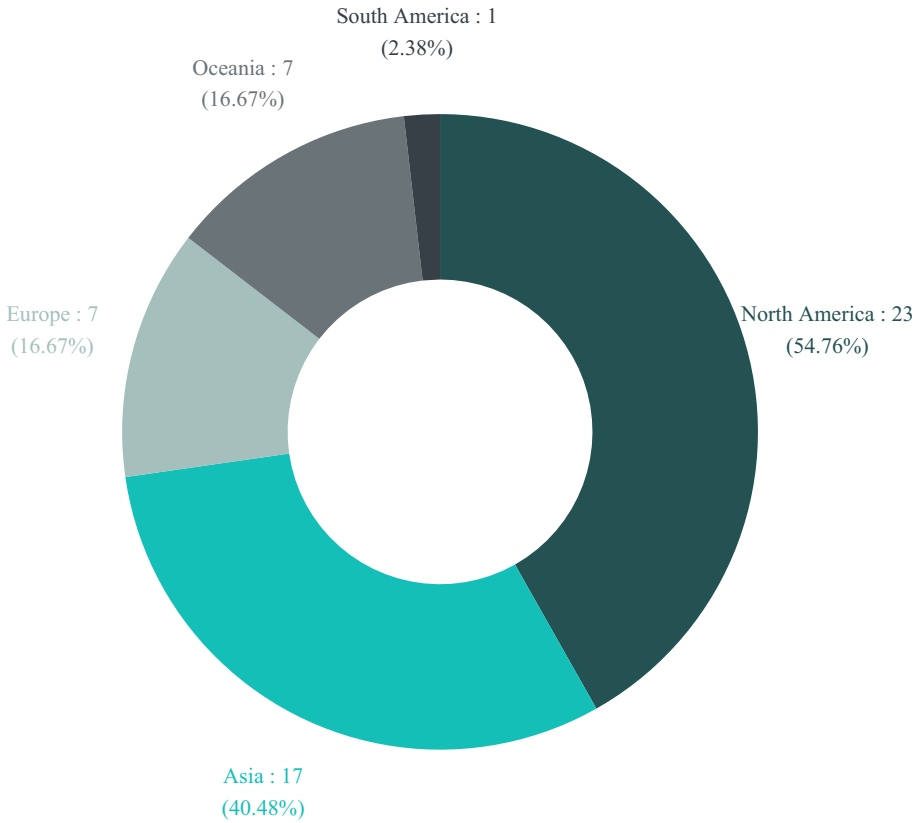

Fig. 3 Number of studies per continent

4.1 Demographics

Figure 2 shows clearly that the majority of the selected studies were published in journals. An increasing trend in publications volume is also present.

The remaining articles were published in conferences with the exception of one which comes from a workshop. As it is possible to observe, only studies published after 2014 made the final stage of this SLR, and almost 65% of them were published in the last 4 years. This provides some indication that, not only SDA is a relatively new practice, but also, that it is becoming mature only in the very last few years of this decade.

Looking in-depth to the publication where the studies appeared, we easily find that the Empirical Software Engineering Journal has a strong dominance among all the others. The distribution of studies per Publication over the Years is presented in Fig. 10 in Appendix 2. Here we can observe that only the Software Quality Journal and the Journal of Systems and Software have more than one study published within our final set of articles.

North America and Asia are the most active regions researching on Software Analytics as plotted previously in Fig. 3. The collaboration between institutions from these two regions is easily detected in the large number of studies that were published in cooperation as can bee seen in Table 12. Canada, Singapore, USA, Australia and China are the most effective countries in producing work in this domain. The most active institutions are also from these countries as we can observe on Fig. 4.

Regarding authorship, which we present the details in Fig. 9 in Appendix 2, we found that only 4 main authors appear with 2 studies in the selected papers, and one of them appear with more than one study per year. All the remaining authors are present with only one publication. This may resonate the difficulty that is to setup, document and publish such type of studies. Figure 4 in Appendix 2, present the frequency of contributions regarding continents, countries and institutions involved, either as primary or secondary authors, on all studies.

4.2 Analysis and Findings

It is widely accepted that we lack experimentation in Software Engineering in general. This phenomenon is even more acute on what concerns experimentation related with analytics in practice for software development. Even if this work is scarce, we should look at it collectively to try to draw some picture of the current state-of-the-art. For that purpose, a summary table with the complete information extracted to

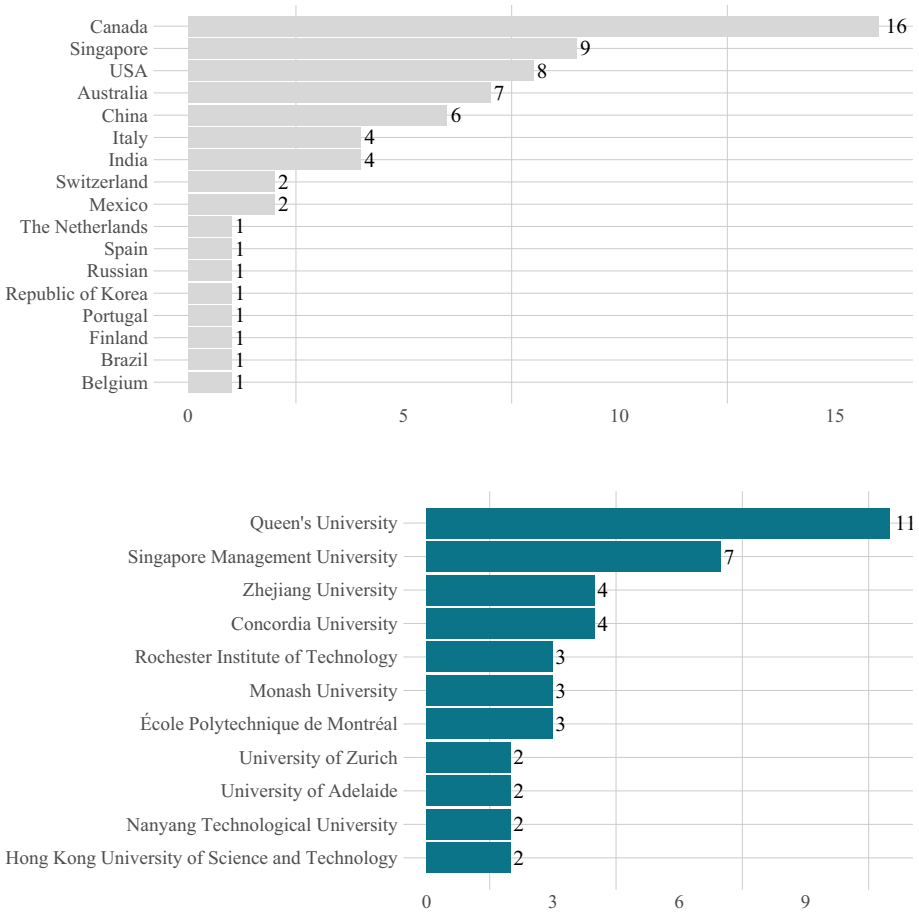

Fig. 4 Number of studies per country and institution (> 1 study only)

answer all the **RQs**, is presented in Table 16 in Appendix 3. In this section we present each research question and the correspondent dimension findings and their frequencies[3].

4.2.1 RQ1. What Type of Empirical Studies have been Conducted?

According to the type of empirical studies provided by [75], from the total number of publications, more than half, 53.12%, are Exploratory Case Studies. Quasi-Experiments and Exploratory Case Studies combined account for 90.62%. This is probably not a surprise, since the remaining study types are, quite often, harder to setup due to technical limitations in the data collection process or blocked by data privacy concerns raised by the entities involved.

One publication, [S13], combines three study types: Exploratory Case Study, Quasi-Experiment and a Survey. Having two types of empirical studies presented, we find [S31] and [S23] which combine a Exploratory Case Study

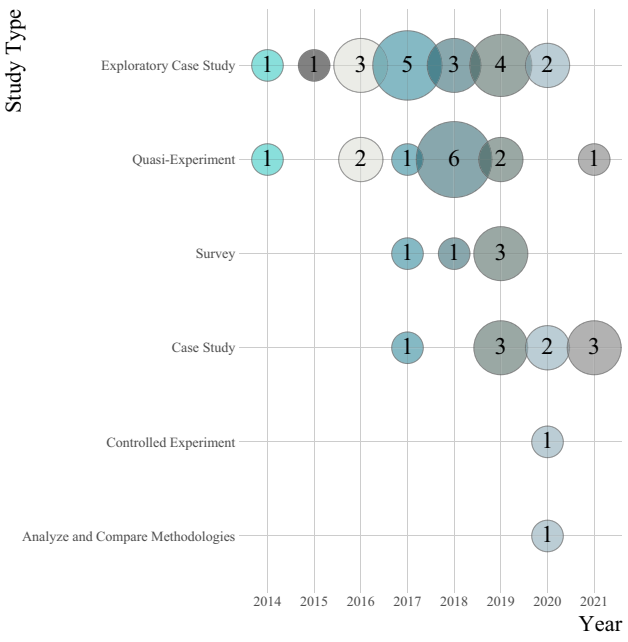

Fig. 5 Frequencies of study types per year

[3] The sum of frequencies might be bigger than the total number of selected studies(n = 42) because some publications have been classified with more than one Study Type, Data Source, SDLC Activity, Stakeholder, Mining Method and/or Analytics Scope.

Table 5 Study type findings

Type	Freq.	%	Ref.
Exploratory case study	19	45.24	[S04], [S05], [S09], [S10], [S13], [S16], [S20], [S22], [S23], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S37], [S40]
Quasi-experiment	13	30.95	[S06], [S07], [S11], [S12], [S13], [S14], [S15], [S17], [S18], [S19], [S21], [S24], [S34]
Case study	9	21.43	[S01], [S02], [S03], [S08], [S33], [S36], [S39], [S41], [S42]
Survey	5	11.9	[S06], [S13], [S23], [S24], [S31]
Analyze and compare methodologies	1	2.38	[S35]
Controlled experiment	1	2.38	[S38]

and a Survey. Having a Quasi-Experiment and a Survey we have [S6] and [S24]. The remaining publications have only one empirical study type given. Study Types found and the plot of their distribution per Year is shown on Fig. 5.

No Meta-Analysis, Experience Report or Discussion had quality to reach the final stage of this SLR. Particularly for the Controlled Experiment studies reduced presence, its worth elaborate that a controlled experiment is one in which all factors are held constant except for one: the independent variable. It is common to compare a control group against an experimental group where all factors are identical between the two groups except for the factor being tested. This approach has the advantage that is easier to eliminate uncertainty about the significance of the results, however, it also has a considerable drawback—the effort needed to design and execute such experiments which may explain partially why there is only one study present in our final list (Table 5).

We believe that sufficient conditions needed to conduct such experiments are not yet being met in software development organizations. Experiments where treatments are applied to some factors in order to later evaluate the outcomes are almost non-existent in real live scenarios. This may reveal that, due to revenue generation pressure, costs control and/or time restrictions, organizations are not willing to spend time and resources to test and experiment novel approaches on analytics even when they promise potential benefits.

RQ1. Highlights

i) Controlled Experiment studies look neglected by the community.
ii) 88.09% (37/42) of works pertain to only one study type (Table 16).
iii) Evidences suggest an increasing trend in the publications quality.

4.2.2 RQ2: What are the Main Data Sources Used for Software Development Related Studies?

The top four data sources: Github Repositories, Google Play Store, Git Repositories and BugZilla combined are the data sources for more than 80% of the studies. This was somehow expected as they are generally under the public domain and contain the code, issue reports and product compilations of the most used open source projects, which are, very often used in empirical studies. This provides some evidence that the community is probably studying the most what is possible to study, simply because the datasets are under the public domain.

Interesting to mention is the high number of publications using datasets from App Stores such as Google Play Store. This might be a relevant indicator that the researchers' focus, the profile of the end-user and the developers' characteristics are quickly and fundamentally changing.

Figure 11, presented in Appendix 2 plots the frequencies of all studies regarding **RQ2**. It is proper to highlight that, from all the data sources used in more than one study, 4 are related with software configuration management systems, 2 with App Stores and each of the remaining 3 with: Bug/Issue Tracking Systems, a Q &A Service and an Online Survey.

RQ2. Highlights

i) Code management and bug/issue tracking systems are used frequently.
ii) App Stores, Q&A services, Wikis and Forums are promising sources.
iii) Repositories containing developers' project interactions are scarce.

Table 6 SDLC activities findings

Activity	Freq.	%	Ref.
Implementation	38	90.48	[S01], [S02], [S04], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S33], [S34], [S35], [S36], [S37], [S40], [S41], [S42]
Maintenance	26	61.9	[S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S17], [S18], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S34], [S35], [S38], [S39], [S40]
Testing	13	30.95	[S01], [S24], [S30], [S33], [S34], [S35], [S36], [S37], [S38], [S39], [S40], [S41], [S42]
Debugging	7	16.67	[S07], [S08], [S09], [S10], [S11], [S12], [S39]
Operations	6	14.29	[S03], [S05], [S18], [S20], [S28], [S35]

Table 7 Stakeholders findings

Stakeholder	Freq.	%	Ref.
Developers	42	100	[S01], [S02], [S03], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S33], [S34], [S35], [S36], [S37], [S38], [S39], [S40], [S41], [S42]
Product managers	17	40.48	[S03], [S06], [S12], [S18], [S20], [S27], [S28], [S33], [S34], [S35], [S36], [S37], [S38], [S39], [S40], [S41], [S42]
Testers	8	19.05	[S01], [S24], [S33], [S34], [S35], [S37], [S40], [S42]
Project managers	3	7.14	[S26], [S29], [S37]
Researchers	3	7.14	[S17], [S20], [S29]
Educators	1	2.38	[S29]
End-users	1	2.38	[S20]
Requirements engineers	1	2.38	[S18]

4.2.3 RQ3: What Type of Process/Project Perspective Analysis was Conducted?

We found that all the studies were focused on a Post-Mortem approach, meaning the study was not designed to help the product/project managers take any corrective measures on a timely manner to the artifact under study. As such, any insights gathered could only impact future developments. A Post-Mortem approach provides benefits for the next product release or project, but usually, not for the one being studied as it brings no added value when proactive corrective actions are desired.

RQ3. Highlights

i) Ineffective approach to improve project under study.
ii) Real-time development operational support is missing.
iii) Worthless approach if project actions recommendation is needed.

4.2.4 RQ4: What are the SDLC Activities Mostly Studied?

According to [27], in Table 6 we summarize which activities of the SDLC, are being researched the most. Our findings show that 90.48% and 61.9% of the studies were targeting the Implementation and Maintenance phases, respectively. Regarding Testing, we found 13 studies. These results, which confirm that some phases are under-researched, require the attention of practitioners and eventually the opening of new streams of investigation on the SDLC. Software under operation was the focus of 6 studies and those were mainly related with software deployed to App Stores. Figure 13 present the statistics about all the activities studied. Table 6 details the activities and summarizes their frequencies and identify the studies on each of them.

RQ4. Highlights

i) More than 90% of articles focus on the analysis of programming activities.
ii) Analytics for software under operation is almost non existing.
iii) Requirements Engineering and Design activities are not being studied.

Table 8 Mining methods findings

Stakeholder	Freq.	%	Ref.
Descriptive statistics	41	97.62	[S01], [S02], [S03], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S33], [S34], [S36], [S37], [S38], [S39], [S40], [S41], [S42]
Correlation analysis	23	54.76	[S01], [S02], [S04], [S05], [S08], [S11], [S14], [S15], [S17], [S18], [S19], [S20], [S21], [S22], [S24], [S25], [S27], [S28], [S32], [S39], [S40], [S41], [S42]
Classifier learning	10	23.81	[S06], [S07], [S08], [S11], [S18], [S21], [S25], [S32], [S40], [S41]
Pattern extraction	9	21.43	[S01], [S02], [S03], [S06], [S07], [S09], [S10], [S13], [S23]
Hyphotesis testing	8	19.05	[S04], [S05], [S11], [S14], [S15], [S17], [S18], [S39]
Analysis	4	9.52	[S33], [S34], [S35], [S38]
Cluster analysis	4	9.52	[S20], [S22], [S25], [S32]
Topic modeling	3	7.14	[S19], [S22], [S29]
Feature extraction	2	4.76	[S08], [S11]
Redundancy analysis	2	4.76	[S08], [S11]
Regression models	2	4.76	[S19], [S20]
Association rules	1	2.38	[S30]
Generalized suffix trees	1	2.38	[S32]
Genetic algorithms	1	2.38	[S29]
Heuristic features	1	2.38	[S07]
Mixed-effect models	1	2.38	[S20]
Natural language processing	1	2.38	[S30]
Process mining	1	2.38	[S16]

4.2.5 RQ5: Who were the Target Stakeholders of These Studies?

All the studies targeted the Developers, and 7 were addressing Product Managers concerns. Only 5 publications could bring any value to Testers: [S01], [S24], Educators: [S29], End-Users: [S20] and Requirements Engineers: [S18]. These findings are aligned with the results found in previous SLRs mentioned in Sect. 2.1. We are predisposed to think that these results are related with the data sources also identified previously. When the majority of data sources used are product code related, it is somehow plausible that the stakeholder for that study is a developer. On summarizing the data about the individuals that could benefit from each study, we argue that the proper insights are not reaching all those who need support on their daily activities, namely Project Managers, Testers and Requirements Engineers. Figure 13 supports our comments by plotting the frequencies of all stakeholders targeted (Table 7, 8).

RQ5. Highlights

i) Developers keep being the main target stakeholder for SDA.
ii) SDA for Testers are less frequent than expected.
iii) High-Level management needs are not being addressed.

4.2.6 RQ6. What are the main mining methods being used?

All articles, as expected, present descriptive statistics about the domain under study. We know that, very often, research starts with just exploratory actions. However, understanding "What happened" is a reduced perspective for what analytics can do for software development. It is also not surprising that the following most frequent methods used are approaches which target the extraction of knowledge, either by correlating factors or by classifying or grouping

Table 9 Analytics scope findings

Scope	Freq.	%	Ref.
Descriptive	42	100	[S01], [S02], [S03], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S33], [S34], [S35], [S36], [S37], [S38], [S39], [S40], [S41], [S42]
Diagnostics	38	90.48	[S01], [S02], [S03], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S27], [S28], [S29], [S30], [S31], [S32], [S34], [S36], [S38], [S39], [S40], [S41]
Predictive	11	26.19	[S06], [S08], [S11], [S18], [S19], [S20], [S21], [S25], [S32], [S39], [S40]
Prescriptive	1	2.38	[S30]

subjects. Hypothesis testing appears less frequently as one would expect. This may be related with the fact that all studies have, as mentioned earlier, a post-mortem approach and any results obtained are not to be used immediately to perform any corrections in the studied project. If used properly, that is what hypothesis testing may bring in advanced forms of analtyics.

Being software development a process, one would expect to find Process Mining methods often in the assessed studies. Looking deep into the data, we can confirm that it does not hold true, which may reveal that practitioners are studying processes without the proper plethora of methods and tools. Figure 12 provide evidences for the most used mining methods.

RQ6. Highlights

i) Few studies try to make any predictions.
ii) Hypothesis Testing appear in only 7(21.88%) of the studies.
iii) Only 1 study (3.12%) used Process Mining methods and tools.

4.2.7 RQ7: Which Type/Form of Analytics was Applied?

Following the rationale in **RQ6**, we found all studies used Descriptive and Diagnostics Analytics together. It makes sense that understanding "hat happened" is complemented with "Why it happened". However, this observation is not fully aligned with the results mentioned in previous SLRs, namely in [47]. Although 28.12% of the studies had some sort of prediction as a goal, that is not reflected in the prescriptive domain, where only 1 study, [S30] aims at suggesting stakeholders actions to improve or correct a development activity. Figure 13 presented in Appendix 2 complements the analysis to this **RQ** (Table 9).

RQ7. Highlights

i) Descriptive and Diagnostics Analytics seems to be found together.
ii) An increasing trend exists in predictive studies (Tables 11 & 16).
iii) Management actions recommendation is not a common practice.

4.2.8 RQ8. What were the relevant contributions to the SDLC?

Technical Debt. All the studies had some sort of contribution to the quality dimension of software and no study was found to be classified with **"Absent"** under this realm. With **"Moderate"** contributions we find [S03], [S22], [S23], [S26], [S28], [S31], [S35], [S38], [S42]. Having a **"Strong"** impact we identify [S01], [S02], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S24], [S25], [S30], [S32], [S36], [S37], [S39], [S40], [S41]. Very few studies have **"Weak"** benefits identified [S21], [S27], [S29], [S33], [S34].

Time Management. The management of project times is analyzed in less than half of the studies since **54.76%** of the studies provide no contribution under this dimension. We identify 11 studies, [S15], [S21], [S26], [S34], [S35], [S36], [S37], [S38], [S39], [S40], [S41] with **"Moderate"** contributions to manage the duration of product/project development. **"Weak"** benefits are present in 8 **(19.05%)** studies [S01], [S02], [S08], [S11], [S18], [S19], [S23], [S30].

Costs Control. A similar scenario happens with the control of costs as only 4 [S34], [S35], [S36], [S37] and 9 studies [S01], [S02], [S04], [S08], [S11], [S21], [S38], [S39], [S40] have **"Moderate"** and **"Weak"** contributions, respectively.

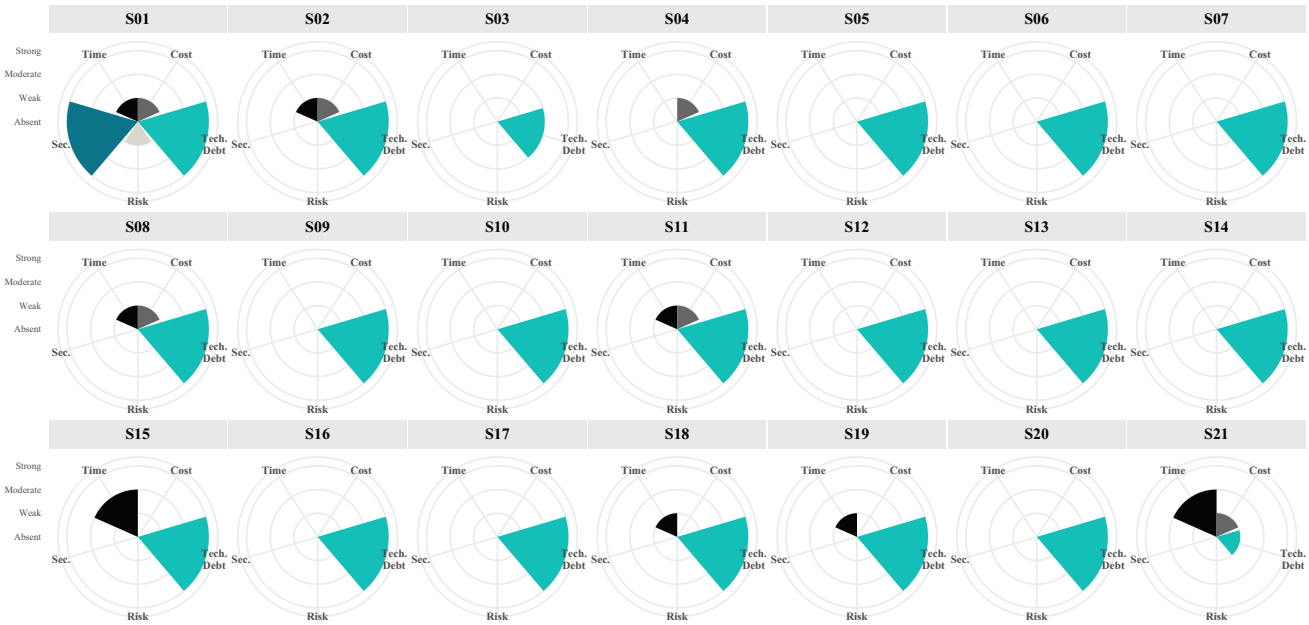

Fig. 6 Classification combining all 5 contribution dimensions to SDLC(RQ8)

Risk Assessment. Despite the fact that risk is cross-cut to all other dimensions identified in **RQ8**, we found only 4 studies, [S01], [S35], [S36], [S37], concerned exactly with the risk associated with the security within the software development process. The contributions given were **"Weak"** though. This means that **90.48%** of the studies did not address at all any concerns involving risk management.

Security Analysis. Regarding software security implementation and operations, we found very few studies where their main contributions were related to this domain. We found 5 studies, [S01], [S27], [S29], [S30] and [S36], where only the first one has a **"Strong"** classification regarding this contribution. Te remaining studies **(88.1%)** did not mention or identified any benefits under this realm.

RQ8. Highlights

i) The software quality dimension consume most research resources.
ii) Time and Costs concerns are not being addressed sufficiently.
iii) Security and Risks matters need extra and aligned effort to evolve.

4.3 Summary

Most of the works focus on the software quality dimension and other features are barely touched by practitioners. Improving or understanding better a project costs, risks and security aspects are contributions rare to find. Only two studies, [S1] and [S36], provide contributions across all the dimensions we assessed and they are essentially **"Moderate"** or **"Weak"** contributions. No study was classified as **"Complete"** on any of the contribution areas identified for the SDLC.

Based on the evidences provided by this study, we observe that **80.9%** (34 out of 42) of the studies were published in Journals, being the Empirical Software Engineering the one with more publications, 24 **(57.1%)**. North America and Asia are the most active regions researching on Software Analytics as plotted previously in Fig. 3. The collaboration between institutions from these two regions is easily detected in the large number of studies that were published in cooperation as can bee seen in Table 12. Canada, Singapore, USA, Australia and China are the most effective countries in producing work in this domain. The most active institutions are also from these countries as we can observe on Fig. 4.

Figure 13, which supports our answers to **RQ1, RQ4, RQ5, RQ7**, plots the frequencies of studies related with the analytics depth, study types, stakeholders and SDLC activities studied.

Figure 6 renders the evaluation off all studies across the five dimensions used to answer **RQ8**. As it is clear from the plots, Technical Debt and Time are the dimensions mostly studied. A list of all studies with a short summary, their context, methods and results are presented in B. A holistic perspective of all the **RQs** findings is presented in C (Fig. 7).

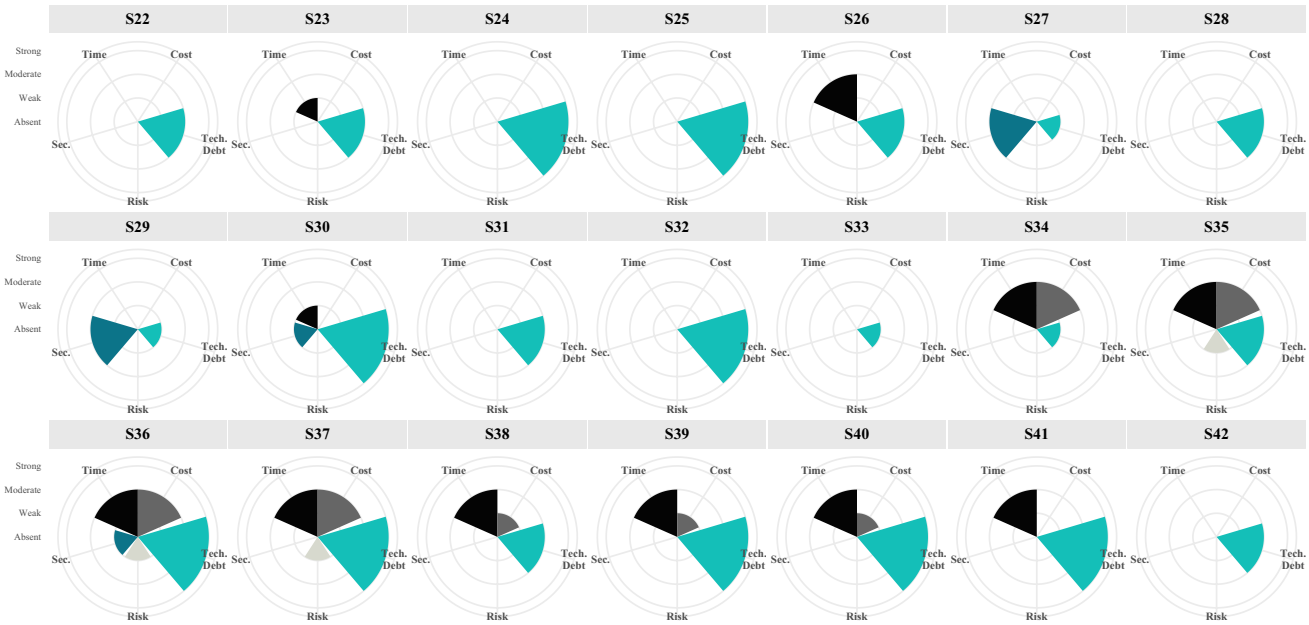

Fig. 7 Classification combining all 5 contribution dimensions to SDLC(RQ8)

4.4 Threats to Validity

The following types of validity issues were considered when interpreting the results from this review.

4.4.1 Construct Validity

The studies identified from the systematic review were accumulated from multiple literature databases covering relevant journals, proceedings and books. One possible threat is bias in the selection of publications. This is addressed through specifying a research protocol that defines the objectives of the study, the research questions, the search strategy and search strings used. Inclusion, exclusion criteria and blueprint for data extraction and quality assessment complements the approach to mitigate such bias.

Although supported by important literature under the software engineering domain, we followed a self-defined classification criteria for some **RQs**, specifically for **RQ8**. This method is somehow subjective as someone else might have chosen any other classification categories.

Our dataset contains studies published until mid July, 2019. There are some evidences pointing to an increasing trend in the publishing of studies in the SDA domain, however, articles published in the second-half of 2019 which might also had good quality, were not included in this review. We excluded works where their goal was only to propose new algorithms and/or methods to analyze software development. Some of these studies had also validation experiments, however, their conclusions were related with the quality of the methods and not with any benefits potentially provided by them for the software development process. Some of those studies had also interesting approaches to improve analytics as a practice, however, they are not present in this review.

4.4.2 Internal Validity

One possible threat is the selection bias and we addressed it during the selection step of the review, i.e. the studies included in this review were identified through a thorough selection process which comprises of multiple stages. We were aiming to find high quality studies, therefore, a quality assessment was introduced and a final selection for studies ranking above the third quartile was conducted. This approach may have excluded studies with very important contributions on any of the dimensions we assessed in **RQ8** or other dimensions not covered by this review. We used an ordinal/categorical taxonomy to assess the studies regarding **RQ8**. This classification method is still subjective and depends on the authors' contents interpretation.

4.4.3 External Validity

There may exist other valid studies on other digital libraries which we did not search. However, we tried to reduce this limitation by exploiting the most relevant software engineering literature repositories. Studies written not in English were excluded which can also have excluded important work which otherwise would have been also mentioned.

4.4.4 Conclusion Validity

There may be bias in the data extraction phase, however, this was addressed through defining a data extraction form to ensure consistent extraction of proper data to answer the research questions. We should also refer that, the findings and further comments are based on this extracted data. Despite the fact that high levels of validation were applied in the statistics computation of this study, there is always a small chance that any figures might be inaccurate. For this reason, we publish our final dataset to enable replication and thus allowing for further validation.

5 Conclusions

We conducted a Systematic Literature Review on SDA in practice, covering a time span between 2010 and 2021. From an initial population of 3,154 papers, we kept 42 of them for appraisal.

It targeted eight specific aspects related with the goals, sources, methods used and contributions provided in certain areas of the SDLC. Our goal was to extract the most relevant dimensions associated with software development practices and highlight where and what were the potential contributions given by those works to the SDLC. From a quality assessment perspective, our aim was also to classify the benefits provided by those studies to significant software development concerns such as: quality/technical debt, time, costs, risks and security, therefore, a taxonomy was created to evaluate them.

Source code repositories, such as GitHub and Git, and App stores like Google Play Store **(top 3 > 50**%), exploratory case studies (**45.24**%), and developers (**100**%) are the most common data sources, study types, and stakeholders, respectively. Testers (**19.05**%) also get moderate attention from researchers. Product managers' (**40.48**%) concerns are being addressed frequently and project managers (**7.14**%) are also present but with less prevalence. Mining methods are rapidly evolving, as reflected in their identified long list. Descriptive statistics (**97.62**%) are the most usual method followed by correlation analysis (**54.76**%). Being software development an important process in every organization, it was unexpected to find that process mining was present in only one study (**2.38**%). Most contributions to the software development life cycle were given in the quality dimension (**100**%). Time management (**45.2**%) and costs control (**30.9**%) were less prevalent. The analysis of security aspects appear in (**11.9**%) of the studies. Although with a small presence in this analysis, evidences suggest it is an increasing topic of concern. Risk management contributions are scarce (**9.52**%).

Our analysis highlighted a number of limitations and shortcomings on the SDA practice and bring the focus to open issues that need to be addressed by future research. It is our understanding, that our work may provide a baseline for conducting future research and the findings presented here will lead to higher quality research in this domain.

5.1 Call for Action

As a final remark and to trigger a call for action in the research community, the following issues should be addressed:

- **Repository Diversity.** We suggest researchers to explore different and non trivial software development related repositories, such as the IDE or other archives containing development events(eg: decisions, fine grain actions executed, etc). More and distinct datasets are expected to expand the analytics coverage on software development.
- **Keep working on the needs of different stakeholders.** We have evidences that the practitioners who benefit most from the current SDA studies are the developers and many other profiles are left behind. We suggest to increase the focus on the real needs of requirements engineers, project, product and portfolio managers and higher level executives.
- **Aim at Software Development Operational Support.** No studies were found providing clear evidences that the outcome of that study could benefit on a timely manner the ongoing project or product versions. If organizations want to focus effectively on detecting, predicting and recommending corrective actions on a timely manner, meaning, any insights gathered will have impact on current project and not solely on the next project or product version, researchers and practitioners should focus on designing advanced tools and methods to address software development operational support.
- **Software Development Process Mining.** Despite the fact that Process Mining is now a mature topic, almost no software process related studies uses it. We suggest its techniques and tools, to study deeper the interaction of software development stakeholders and to complement the effectiveness of assessing certain software development tasks, such as, project effort prediction, code maintenance activities and/or bug detection methods.
- **Project Time and Costs.** We suggest more and deeper studies covering the Time and Costs of software projects. These are dimensions barely addressed by the studies we evaluated. The aforementioned topics are extremely relevant to forecast resource allocation for future projects.
- **Address Security and Risks holistically.** Due to the unceasing digital transformation present nowadays in the

society, the security of information systems will be even more critical to any organization. We now have robust methods to assess security vulnerabilities in software code. However, very little is known about the developers behaviour during the Implementation and Maintenance phases, just to name a few. Even if, in the last years, security in general became quickly a pertinent topic, the security around development processes and the involved resources are still not clearly addressed. This is a topic with increasing relevance and deserves the rapid and focused attention from the practitioners.

– **Blockchain.** One of the most interesting, promising and relevant technological contributions to the society, was created roughly ten years ago - the birth of *bitcoin* [46]. Although *bitcoin* is an implementation of electronic money, it is supported by something very powerful, which can be used for many other use cases, called - *blockchain* [60]. The *blockchain* is a mechanism which is able to keep a book of data records immutable and distributed across a multi-node network of servers. It is virtually indestructible since it has no central authority controlling it and preserves data integrity by potentially not allowing rollback on any past transactions. Additionally, if required, it guarantees that only the data owners are able to view or change their personal records and yet permit third-parties to be granted view only privileges to a selected dataset. This technology may be used embedded in SDA to anonymize and grant privacy to organizations sharing data without spoil the context associated with the development process under study.

Appendices

Appendix 1: Data Extraction

Selection Process

See Fig. 8, Table 10.

Appendix 2: Studies List

See Table 11.

A Data Extraction

Selection Process

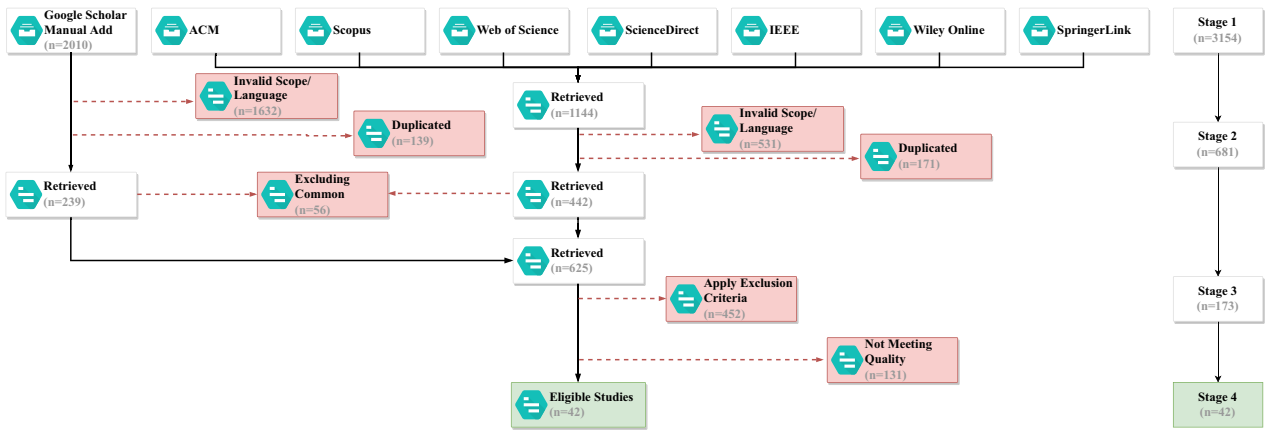

Fig. 8 Study selection process stages

Table 10 Data collection form

#	Item	Description
General Information		
1	Publication Id	A sequential identifier for each publication
2	Extraction date	Date/Time when the data was extracted
3	Bibliography reference	The references of each publication
4	Publication date	The Date/Time of publishing
5	Publication type	The type publication (e.g. Journal, Conference, etc)
6	Publisher name	The name of the publisher
7	Publication Author(s)	The author(s) of the publication
Addressing RQs		
8	Study Type(s)	Extracting the type of empirical study as defined in [42, 75]
9	Data source(s)	The different types of data sources used in the publications. Admissible values are open
10	Process perspective	The timing of when the study was conducted (e.g. **Pre-Mortem**, if study was executed before project/product was finished, **Post-Mortem**, if it was conducted after)
11	SDLC Activity(ies)	We followed the SWEBOK to build our list of admissible values [27]. **Implementation** - refers to the activity of constructing artifacts for a new product based on new defined requirements and design. **Maintenance** - refers to the task of maintaining, by changing or evolving an existing software under operation according to early defined specifications. **Testing** - refers to the automated or manual task of finding bugs and/or errors. **Debugging** - is the effort of fixing those known bugs. **Operations** - is related with the phase where the software is under exploration by the end-users. Our approach extends the taxonomy used by [15]
12	Study Stakeholder(s)	The publication outcomes should be targeted to specific individuals in the software development process. We identify them here
13	Mining Method(s)	The identification of the methods used for data mining/analysis
14	Analytics Scope(s)	Identifies what type of analytics was performed. We used the valid options(**Descriptive Analytics**, **Diagnostics Analytics**, **Predictive Analytics** and **Prescriptive Analytics**) identified in [16]
15	Contribution(s) to SDLC	We framed the admissible options to the following assessment dimensions of software: **Technical Debt/Quality**, **Time**, **Costs**, **Risk** and **Security**. Our approach adapt and extends some of the dimensions and concerns identified earlier in Sect. 2.1
Findings		
16	Findings and conclusions	What were the interpretation of the results obtained
17	Validity	Identifying the threats to the validity of the publication
18	Relevance	What other relevant outcomes could be inferred from the publication other then the ones in item 15

Comments on Studies

[**S01**] explores the correlation between software vulnerabilities and code-level constructs called micro patterns. The authors analyzed the correlation between vulnerabilities and micro patterns from different viewpoints and explored whether they are related. The conclusion shows that certain micro patterns are frequently present in vulnerable classes and that there is a high correlation between certain patterns that coexist in a vulnerable class [58].

[**S02**] presents an empirical study to analyze commit histories of Android manifest files of hundreds of apps to understand their evolution through configuration changes. The results is a contribution to help developers in identifying change-proneness attributes, including the reasons behind the changes and associated patterns and understanding the usage of different attributes introduced in different versions of the Android platform. In summary, the results show that most of the apps extend core functionalities and improve user interface over time. It detected that significant effort is wasted in changing configuration and then reverting back the change, and that very few apps adopt new attributes introduced by the platform and when they do, they are slow in adopting new attributes. Configuration changes are mostly influenced by functionalities extension, platform evolution and bug reports [29].

[**S03**] studied updates in the Google Play Store by examining more than 44,000 updates of over 10,000 mobile apps, from where 1,000 were identified as emergency updates. After studying the characterirstics of the updates, the authors found that the emergency updates often have a long lifetime (i.e., they are rarely followed by another emergency update) and that updates preceding emergency updates often

Table 11 Systematic literature review studies

#	Score	Year	Author	Title	Publication
S01	12	2019	Sultana et al.	A study examining relationships between micro patterns and security vulnerabilities	Software Quality Journal
S02	10.5	2019	Jha et al.	An empirical study of configuration changes and adoption in Android apps	Journal of Systems and Software
S03	10	2017	Hassan et al.	An empirical study of emergency updates for top android mobile apps	Empirical Software Engineering
S04	10	2016	W. Wu et al.	An exploratory study of api changes and usages based on apache and eclipse ecosystems	Empirical Software Engineering
S05	10.5	2017	Taba et al.	An exploratory study on the usage of common interface elements in android applications	Journal of Systems and Software
S06	10	2019	Prana et al.	Categorizing the Content of GitHub README Files	Empirical Software Engineering
S07	10	2018	R. Wu et al.	ChangeLocator: locate crash-inducing changes based on crash reports	Empirical Software Engineering
S08	10.5	2019	Yan et al.	Characterizing and identifying reverted commits	Empirical Software Engineering
S09	10	2018	Salza et al.	Do developers update third-party libraries in mobile apps?	International Conference on Program Comprehension
S10	10	2019	Liu et al.	DroidLeaks: a comprehensive database of resource leaks in Android apps	Empirical Software Engineering
S11	11	2018	Fan et al.	Early prediction of merged code changes to prioritize reviewing tasks	Empirical Software Engineering
S12	10	2016	McIlroy et al.	Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store	Empirical Software Engineering
S13	11	2018	Saborido et al.	Getting the most from map data structures in Android	Empirical Software Engineering
S14	10	2017	Guerrouj et al.	Investigating the relation between lexical smells and change- and fault-proneness: an empirical study	Software Quality Journal
S15	10	2014	Fucci et al.	On the role of tests in test-driven development: a differentiated and partial replication	Empirical Software Engineering
S16	10	2014	Mittal et al.	Process mining software repositories from student projects in an undergraduate software engineering course	International Conference on Software Engineering
S17	10.5	2018	Rakha et al.	Revisiting the performance of automated approaches for the retrieval of duplicate reports in issue tracking systems that perform just-in-time duplicate retrieval	Empirical Software Engineering
S18	10	2018	Morales-Ramirez et al.	Speech-acts based analysis for requirements discovery from online discussions	Information Systems Journal
S19	10.5	2018	Li et al.	Studying software logging using topic models	Empirical Software Engineering
S20	11	2018	Hassan et al.	Studying the dialogue between users and developers of free apps in the Google Play Store	Empirical Software Engineering
S21	10	2016	Rakha et al.	Studying the needed effort for identifying duplicate issues	Empirical Software Engineering
S22	10.5	2017	Ye et al.	The structure and dynamics of knowledge network in domain-specific Q &A sites: a case study of stack overflow	Empirical Software Engineering
S23	10.5	2019	Sawant et al.	To react, or not to react: Patterns of reaction to API deprecation	Empirical Software Engineering
S24	10	2019	Cruz et al.	To the attention of mobile software developers: guess what, test your app!	Empirical Software Engineering
S25	10	2017	Li et al.	Towards just-in-time suggestions for log changes	Empirical Software Engineering
S26	10	2017	Izquierdo-Cortazar et al.	Using Metrics to track code review performance	International Conference on Evaluation and Assessment in Software Engineering

Table 11 (continued)

#	Score	Year	Author	Title	Publication
S27	10	2016	Munaiah et al.	Vulnerability severity scoring and bounties: Why the disconnect?	International Workshop on Software Analytics
S28	10	2015	Tian et al.	What are the characteristics of high-rated apps? A case study on free Android Applications	International Conference on Software Maintenance and Evolution
S29	11	2016	Yang et al.	What security questions do developers ask? a large-scale study of stack overflow posts	Journal of Computer Science and Technology
S30	10.5	2019	Chen et al.	What's Spain's Paris ? Mining analogical libraries from Q &A discussions	Empirical Software Engineering
S31	10	2017	Jiang et al.	Why and how developers fork what from whom in GitHub	Empirical Software Engineering
S32	10.5	2019	Thongtanunam et al.	Will this clone be short-lived? Towards a better understanding of the characteristics of short-lived clones	Empirical Software Engineering
S33	11	2020	Avila et al.	A Data Driven Platform for Improving Performance Assessment of Software Defined Storage Solutions	Advances in Intelligent Systems and Computing
S34	12	2021	Wani et al.	A Generic Analogy-Centered Software Cost Estimation Based on Differential Evolution Exploration Process	Computer Journal
S35	12	2020	Rana et al.	A Study of Hyper-Parameter Tuning in the Field of Software Analytics	International Conference on Electronics, Communication and Aerospace Technology
S36	12	2021	Vashisht et al.	An empirical study of heterogeneous cross-project defect prediction using various statistical techniques	International Journal of e-Collaboration
S37	10.5	2020	Capizza et al.	Anomaly Detection in DevOps Toolchain	International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment
S38	11	2020	Avila et al.	Effects of contextual information on maintenance effort: A controlled experiment	Journal of Systems and Software
S39	12	2021	Qu et al.	Evaluating network embedding techniques performances in software bug prediction	Empirical Software Engineering
S40	11	2020	Krishna et al.	Learning actionable analytics from multiple software projects	Empirical Software Engineering
S41	10	2020	Bangash et al.	On the time-based conclusion stability of cross-project defect prediction models	Empirical Software Engineering
S42	10	2021	AIOmar et al.	Toward the automatic classification of Self-Affirmed Refactoring	Journal of Systems and Software

receive a higher ratio of negative reviews than the emergency updates [25].

[**S04**] analyzed and classified API changes and usages together in 22 framework releases from the Apache and Eclipse ecosystems and their client programs. The authors conclude that missing classes and methods happen more often in frameworks and affect client programs more often than the other API change types do, and that missing interfaces occur rarely in frameworks but affect client programs often. In summary, framework APIs are used on average in 35% of client classes and interfaces and most of such usages could be encapsulated locally and reduced in number. Around 11% of APIs usages could cause ripple effects in client programs when these APIs change. Some suggestions for developers and researchers were made to mitigate the impact of API evolution through language mechanisms and design strategies [70].

[**S05**] extracted commonly used UI elements, denoted as Common Element Sets (CESs), from user interfaces of applications. The highlight the characteristics of CESs that can result in a high user-perceived quality by proposing various metrics. From an empirical study on 1292 mobile applications, the authors observed that CESs of mobile applications widely occur among and across different categories, whilst certain characteristics of CESs can provide a high user-perceived quality. A recommendation is made, aiming to improve the quality of mobile applications, consisting on the adoption of reusable UI templates that are extracted and summarized from CESs for developers [59].

[**S06**] performed a qualitative study involving the manual annotation of 4,226 README file sections from 393 randomly sampled GitHub repositories and design and evaluate a classifier and a set of features that can categorize these sections automatically. The findings show that information discussing the 'What' and 'How' of a repository hapens very often, while at the same time, many README files lack information regarding the purpose and status of a repository. A classifier was built to predict multiple categories and the F1 score obtained encourages its usage by software repositories owners. The approach presented is said to improve the quality of software repositories documentation and it has the potential to make it easier for the software development community to discover relevant information in GitHub README files [49].

[**S07**] conducted an empirical study on characterizing the bug inducing changes for crashing bugs (denoted as crash-inducing changes). ChangeLocator was also proposed as a method to automatically locate crash-inducing changes for a given bucket of crash reports. The study approach is based on a learning model that uses features originated from the empirical study itself and a model was trained using the data from the historical fixed crashes. ChangeLocator was evaluated with six release versions of the Netbeans project. The analysis and results show that it can locate the crash-inducing changes for 44.7%, 68.5%, and 74.5% of the bugs by examining only top 1, 5 and 10 changes in the recommended list, respectively, which is said to outperform other approaches [69].

[**S08**] explored if one can characterize and identify which commits will be reverted. The authors characterized commits using 27 commit features and build an identification model to identify commits that will be reverted. Reverted commits were identified by analyzing commit messages and comparing the changed content, and extracted 27 commit features that were divided into three dimensions: change, developer and message. An identification model (e.g., random forest) was built and evaluated on an empirical study on ten open source projects including a total of 125,241 commits. The findings show that the 'developer' is the most discriminative dimension among the three dimensions of features for the identification of reverted commits. However, using all the three dimensions of commit features leads to better performance of the created models [71].

[**S09**] conducted an empirical study on the evolution history of almost three hundred mobile apps, by investigating whether mobile developers actually update third-party libraries, checking which are the categories of libraries with respect to the developers' proneness to update their apps, looking for what are the common patterns followed by developers when updating a software library, and whether high- and low-rated apps present any particular update patterns. Results showed that mobile developers rarely update their apps with respect to the used libraries, and when they do, they mainly tend to update the libraries related to the Graphical User Interface, with the aim of keeping the mobile apps updated with the latest design trends. In some cases developers ignore updates because of a poor awareness of the benefits, or a too high cost/benefit ratio [56].

[**S10**] extracted real resource leak bugs from a bug database named DROIDLEAKS. It consisted in mining 34 popular open-source Android apps, which resulted in a dataset having a total of 124,215 code revisions. After filtering and validating the data, the authors found, on 32 analyzed apps, 292 fixed resource leak bugs, which cover a diverse set of resource classes. To fully comprehend these bugs, they performed an empirical study, which revealed the characteristics of resource leaks in Android apps and common patterns of resource management mistakes made by developers [36].

[**S11**] built a merged code change prediction tool leveraging machine learning techniques, and extracted 34 features from code changes, which were grouped into 5 dimensions: code, file history, owner experience, collaboration network, and text. Experiments were executed on three open source projects (i.e., Eclipse, LibreOffice, and OpenStack), containing a total of 166,215 code changes. Across three datasets, the results show statistically significantly improvements in detecting merged code changes and in distinguishing important features on merged code changes from abandoned ones [20].

[**S12**] studied the frequency of updates of 10,713 mobile apps (the top free 400 apps at the start of 2014 in each of the 30 categories in the Google Play store). It was found that only ~1% of the studied apps are updated at a very frequent rate - more than one update per week and 14% of the studied apps are updated on a bi-weekly basis (or more frequently). Results also show that 45% of the frequently-updated apps do not provide the users with any information about the rationale for the new updates and updates exhibit a median growth in size of 6%. The authors conclude that developers should not shy away from updating their apps very frequently, however the frequency should vary across store categories. It was observed that developers do not need to be too concerned about detailing the content of new updates as it appears that users are not too concerned about such information and, that users highly rank frequently-updated apps instead of being annoyed about the high update frequency [37].

[**S13**] studied the use of map data structure implementations by Android developers and how that relates with saving CPU, memory, and energy as these are major concerns of users wanting to increase battery life. The authors initially performed an observational study of 5713 Android apps in GitHub and then conducted a survey to assess developers' perspective on Java and Android map implementations. Finally, they performed an experimental study comparing

HashMap, ArrayMap, and SparseArray variants map implementations in terms of CPU time, memory usage, and energy consumption. The conclusions provide guidelines for choosing among the map implementations: HashMap is preferable over ArrayMap to improve energy efficiency of apps, and SparseArray variants should be used instead of HashMap and ArrayMap when keys are primitive types [55].

[**S14**] detected 29 smells consisting of 13 design smells and 16 lexical smells in 30 releases of three projects: ANT, ArgoUML, and Hibernate. Further, the authors analyzed to what extent classes containing lexical smells have higher (or lower) odds to change or to be subject to fault fixing than other classes containing design smells. The results obtained bring empirical evidence on the fact that lexical smells can make, in some cases, classes with design smells more fault-prone. In addition, it was empirically demonstrated that classes containing design smells only are more change- and fault-prone than classes with lexical smells only [24].

[**S15**] examined the nature of the relationship between tests and external code quality as well as programmers' productivity in order to verify/refute the results of a previous study. With the focus on the role of tests, a differentiated and partial replication of the original study and related analysis was conducted. The replication involved 30 students, working in pairs or as individuals, in the context of a graduate course, and resulted in 16 software artifacts developed. Significant correlation was found between the number of tests and productivity. No significant correlation found between the number of tests and external code quality. For both cases we observed no statistically significant interaction caused by the subject units being individuals or pairs. Results obtained are consistent with the original study although, as the authors admit, there were changes in the timing constraints for finishing the task and the enforced development processes [21].

[**S16**] presented an application of mining three software repositories: team wiki (used during requirement engineering), version control system (development and maintenance) and issue tracking system (corrective and adaptive maintenance) in the context of an undergraduate Software Engineering course. Visualizations, metrics and algorithms to provide an insight into practices and procedures followed during various phases of a software development life-cycle were proposed and these provided a multi-faceted view to the instructor serving as a feedback tool on development process and quality by students. Event logs produced by software repositories were mined and derived insights such as degree of individual contributions in a team, quality of commit messages, intensity and consistency of commit activities, bug fixing process trend and quality, component and developer entropy, process compliance and verification. Experimentation revealed that not only product but process quality varies signicantly between student teams and mining process aspects can help the instructor in giving directed and specific feedback. Authors, observed that commit patterns characterizing equal and un-equal distribution of workload between team members, patterns indicating consistent activity in contrast to spike in activity just before the deadline, varying quality of commit messages, developer and component entropy, variation in degree of process compliance and bug fixing quality [41].

[**S17**] investigated the impact of the just-in-time duplicate retrieval on the duplicate reports that end up in the ITS of several open source projects, namelly Mozilla-Firefox, Mozilla-Core and Eclipse-Platform. The differences between duplicate reports for open source projects before and after the activation of this new feature were studied. Findings showed that duplicate issue reports after the activation of the just-in-time duplicate retrieval feature are less textually similar, have a greater identification delay and require more discussion to be retrieved as duplicate reports than duplicates before the activation of the feature [52].

[**S18**] exploited a linguistic technique based on speech-acts for the analysis of online discussions with the ultimate goal of discovering requirements-relevant information. The datasets used in the experimental evaluation, which are publicly available, were taken from a widely used open source software project (161120 textual comments), as well as from an industrial project in the home energy management domain. The approach used was able to successfully classify messages into Feature/Enhancement and Other, with significant accuracy. Evidence was found to support the rationale, that there is an association between types of speech-acts and categories of issues, and that there is correlation between some of the speechacts and issue priority, which could open other streams of research [44].

[**S19**] studied the relationship between the topics of a code snippet and the likelihood of a code snippet being logged (i.e., to contain a logging statement). The intuition driving this research, was that certain topics in the source code are more likely to be logged than others. To validate the assumptions a case study was conducted on six open source systems. The analysis gathered evidences that i) there exists a small number of "log-intensive" topics that are more likely to be logged than other topics; ii) each pair of the studied systems share 12% to 62% common topics, and the likelihood of logging such common topics has a statistically significant correlation of 0.35 to 0.62 among all the studied systems. In summary, the findings highlight the topics containing valuable information that can help guide and drive developers' logging decisions [35].

[**S20**] revisits a previous work in more depth by studying 4.5 million reviews with 126,686 responses for 2,328 top free-to-download apps in the Google Play Store. One of the major findings is that the assumption that reviews are

static is incorrect. In particular, it is found that developers and users in some cases use this response mechanism as a rudimentary user support tool, where dialogues emerge between users and developers through updated reviews and responses. In addition, four patterns of developers were identified: 1) developers who primarily respond to only negative reviews, 2) developers who primarily respond to negative reviews or to reviews based on their contents, 3) developers who primarily respond to reviews which are posted shortly after the latest release of their app, and 4) developers who primarily respond to reviews which are posted long after the latest release of their app. To perform a qualitative analysis of developer responses to understand what drives developers to respond to a review, the authors analyzed a statistically representative random sample of 347 reviews with responses for the top ten apps with the highest number of developer responses. Seven drivers that make a developer respond to a review were identified, of which the most important ones are to thank the users for using the app and to ask the user for more details about the reported issue. In summary, there were significant evidences found, that it can be worthwhile for app owners to respond to reviews, as responding may lead to an increase in the given rating and that studying the dialogue between user and developer can provide valuable insights which may lead to improvements in the app store and the user support process [26].

[**S21**] empirically examined the effort that is needed for manually identifying duplicate reports in four open source projects, i.e., Firefox, SeaMonkey, Bugzilla and Eclipse-Platform. Results showed that: (i) More than 50% of the duplicate reports are identified within half a day. Most of the duplicate reports are identified without any discussion and with the involvement of very few people; (ii) A classification model built using a set of factors that are extracted from duplicate issue reports classifies duplicates according to the effort that is needed to identify them with significant values for precision, recall and ROC area; and (iii) Factors that capture the developer awareness of the duplicate issues' peers (i.e., other duplicates of that issue) and textual similarity of a new report to prior reports are the most influential factors found. The results highlight the need for effort-aware evaluation of approaches that identify duplicate issue reports, since the identification of a considerable amount of duplicate reports (over 50%) appear to be a relatively trivial task for developers. As a conclusion, the authors highlight the fact that, to better assist developers, research on identifying duplicate issue reports should put greater emphasis on assisting developers in identifying effort-consuming duplicate issues [51].

[**S22**] analyzed URL sharing activities in Stack Overflow. The approach was to use open coding method to analyze why users share URLs in Stack Overflow, and develop a set of quantitative analysis methods to study the structural and dynamic properties of the emergent knowledge network in Stack Overflow. The findings show: i) Users share URLs for diverse categories of purposes. ii) These URL sharing behaviors create a complex knowledge network with high modularity, assortative mixing of semantic topics, and a structure skeleton consisting of highly recognized knowledge units. iii) The structure of the knowledge network with respect to indegree distribution is scale-free (i.e., stable), in spite of the ad-hoc and opportunistic nature of URL sharing activities, while the outdegree distribution of the knowledge network is not scale-free. iv) The indegree distributions of the knowledge network converge quickly, with small changes over time after the convergence to the stable distribution. The conclusions highlight the fact that the knowledge network is a natural product of URL sharing behavior that Stack Overflow supports and encourages, and proposed an explanatory model based on information value and preferential attachment theories to explain the underlying factors that drive the formation and evolution of the knowledge network in Stack Overflow [74].

[**S23**] questioned if there was really a strong argument for the Java 9 language designers to change the implementation of the deprecation warnings feature after they notice no one was taking seriously those and continued using outdated features. The goal was to start by identifying the various ways in which an API consumer can react to deprecation and then to create a dataset of reaction patterns frequency consisting of data mined from 50 API consumers totalling 297,254 GitHub based projects and 1,322,612,567 type-checked method invocations. Findings show that predominantly consumers do not react to deprecation and a survey on API consumers was done to try to explain this behavior and by analyzing if the APIs deprecation policy had an impact on the consumers' decision to react. The manual inspection of usages of deprecated API artifacts lead to the discovery of six reaction patterns. Only 13% of API consumers update their API versions and 88% of reactions to deprecation is doing nothing. However the survey got a different result, where 69% of respondents say they replace it with the recommended repalcement. Over 75% of the API barelly affect consumers with deprecation and 15% of the consumers are affected only by 2 APIs(hibernate-core and mongo-java-driver) [57].

[**S24**] investigated working habits and challenges of mobile software developers with respect to testing. A key finding of this exhaustive study, using 1000 Android apps, demonstrates that mobile apps are still tested in a very ad hoc way, if tested at all. However, it is shown that, as in other types of software, testing increases the quality of apps (demonstrated in user ratings and number of code issues). Furthermore, there is evidence that tests are essential when it comes to engaging the community to contribute to mobile open source software. The authors discuss reasons and

potential directions to address the findings. Yet another relevant finding of this study is that Continuous Integration and Continuous Deployment (CI/CD) pipelines are rare in the mobile apps world (only 26% of the apps are developed in projects employing CI/CD) - authors argue that one of the main reasons is due to the lack of exhaustive and automatic testing [14].

[**S25**] tries to understand the reasons for log changes and, proposes an approach that can provide developers with log change suggestions as soon as they commit a code change, which is referred to as "just-in-time" suggestions for log changes. A set of measures is derived based on manually examining the reasons for log changes and individual experiences. Those measures were used as explanatory variables in random forest classifiers to model whether a code commit requires log changes. These classifiers can provide just-in-time suggestions for log changes and was evaluated with a case study on four open source projects: Hadoop, Directory Server, Commons HttpClient, and Qpid. Findings show that: i) the reasons for log changes can be grouped along four categories: block change, log improvement, dependence-driven change, and logging issue; ii) the random forest classifiers can effectively suggest whether a log change is needed; iii) the characteristics of code changes in a particular commit and the current snapshot of the source code are the most influential factors for determining the likelihood of a log change in a commit [34].

[**S26**] designed and conducted, with the continuous feedback of the Xen Project Advisory Board, a detailed analysis focused on finding problems associated with the large increase over time in the number of messages related to code review. The increase was being perceived as a potential signal of problems with their code review process and the usage of metrics was suggested to track the performance of it. As a result, it was learned how in fact the Xen Project had some problems, but at the moment of the analysis those were already under control. It was found as well how diferent the Xen and Netdev projects were behaving with respect to code review performance, despite being so similar from many points of view. A comprehensive methodology, fully automated, to study Linux-style code review was proposed [28].

[**S27**] analyzed the Common Vulnerability Scoring System (CVSS) scores and bounty awarded for 703 vulnerabilities across 24 products. CVSS is the de facto standard for vulnerability severity measurement today and is crucial in the analytics driving software fortification. It was found a weak correlation between CVSS scores and bounties, with CVSS being more likely to underestimate bounty. Such a negative result is suggested to be a cause for concern. The authors, investigated why the measurements were so discordant by i) analyzing the individual questions of CVSS with respect to bounties and ii) conducting a qualitative study to find the similarities and diferences between CVSS and the publicly-available criteria for awarding bounties. It was found that the bounty criteria were more explicit about code execution and privilege escalation whereas CVSS makes no explicit mention of those. Another lesson learnt was that bounty valuations are evaluated solely by project maintainers, whereas CVSS has little provenance in practice [45].

[**S28**] through a case study on 1,492 high-rated and low-rated free apps mined from the Google Play store, investigated 28 factors along eight dimensions to understand how high-rated apps are different from low-rated apps. The search for the most influential factors was also addressed by applying a random-forest classifier to identify high-rated apps. The results show that high-rated apps are statistically significantly different in 17 out of the 28 factors that we considered. The experiment also presents eveidences for the fact that the size of an app, the number of promotional images that the app displays on its web store page, and the target SDK version of an app are the most influential factors [62].

[**S29**] conducted a large-scale study on security-related questions on Stack Overflow. Two heuristics were used to extract from the dataset the questions that are related to security based on the tags of the posts. Later, to cluster different security-related questions based on their texts, an advanced topic model, Latent Dirichlet Allocation (LDA) tuned using Genetic Algorithm (GA) was used. Results show that security-related questions on Stack Overflow cover a wide range of topics, which belong to five main categories: web security, mobile security, cryptography, software security, and system security. Among them, most questions are about web security. In addition, it was found that the top four most popular topics in the security area are "Password", "Hash", "Signature" and "SQL Injection", and the top eight most difficulty security-related topics are "JAVA Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password", suggesting these are the ones in need for more attention [72].

[**S30**] present an approach to recommend analogical libraries based on a knowledge base of analogical libraries mined from tags of millions of Stack Overflow questions. The approach was implemented in a proof-of-concept web application and more than 34.8 thousands of users visited the website from November 2015 to August 2017. Results show evidences that accurate recommendation of analogical libraries is not only possible but also a desirable solution. Authors validated the usefulness of their analogical-library recommendations by using them to answer analogical-library questions in Stack Overflow [12].

[**S31**] explored why and how developers fork what from whom in GitHub. This approach was supported by collecting a dataset containing 236,344 developers and 1,841,324 forks. It was also validated by a survey in order to analyze the programming languages and owners of forked repositories. Among the main findings we have: i) Developers

fork repositories to submit pull requests, fix bugs, add new features and keep copies etc. Developers find repositories to fork from various sources: search engines, external sites (e.g., Twitter, Reddit), social relationships, etc. More than 42% of developers that were surveyed agree that an automated recommendation tool is useful to help them pick repositories to fork, while more than 44.4% of developers do not value a recommendation tool. Developers care about repository owners when they fork repositories. ii) A repository written in a developers' preferred programming language is more likely to be forked. iii) Developers mostly fork repositories from creators. In comparison with unattractive repository owners, attractive repository owners have higher percentage of organizations, more followers and earlier registration in GitHub. The results show that forking is mainly used for making contributions of original repositories, and it is beneficial for OSS community. In summary, there is evidence of the value of recommendation and provide important insights for GitHub to recommend repositories [30].

[**S32**] designed and executed an empirical study on six open source Java systems to better understand the life expectancy of clones. A random forest classifier was built with the aim of determining the life expectancy of a newly-introduced clone (i.e., whether a clone will be short-lived or longlived) and it was confimed to have good accuracy on that task. Results show that a large number of clones (i.e., 30% to 87%) lived in the systems for a short duration. Moreover, it finds that although short-lived clones were changed more frequently than long-lived clones throughout their lifetime, short-lived clones were consistently changed with their siblings less often than long-lived clones. Findings show that the churn made to the methods containing a newly-introduced clone, the complexity and size of the methods containing the newly- introduced clone are highly influential in determining whether the newly-introduced clone will be short-lived. Furthermore, the size of a newly-introduced clone shares a positive relationship with the likelihood that the newly introduced clone will be short-lived. Results suggest that, to improve the efficiency of clone management efforts, such as the planning of the most effective use of their clone management resources in advance, practitioners can leverage the presented classifiers and insights in order to determine the life expectancy of clones [61].

[**S33**] This paper introduces DDP (Data Driven Plataform) platform, a scalable platform to analyze and exploit performance data. This platform centralizes, analyzes and visualizes the performance data produced during the software development cycle. DDP employs big data and analytics technology to collect, store and process performance data in an efficient and integrated way. They have demonstrated the successful application of DDP for Spectrum Scale, a software defined storage solution, where they have been able to implement performance regression data analysis to validate the performance consistency of new produced builds [4].

[**S34**] To help the industry practitioners in these situations, a analogy-centered model based on differential evolution exploration process is proposed in this research study. The proposed model has been assessed on 676 projects from 5 different data sets and the results achieved are significantly better when compared with other benchmark analogy-based estimation studies [67].

[**S35**] The paper attempts to analyze and compare various methodologies to tune the defect predictors. The research papers which are analyzed here have used data-set from the PROMISE repository, open-source [53].

[**S36**] This paper evaluates empirically and theoretically heterogeneous Cross-project defect prediction (HCPDP) modeling, which comprises of three main phases: Feature ranking and feature selection, metric matching, and finally, predicting defects in the target application. The research work has been experimented on 13 benchmarked datasets of three open source projects. Results show that performance of HCPDP is very much comparable to baseline within project defect prediction [66].

[**S37**] An anomaly detection system can operate in the staging environment to compare the current incoming release with previous ones according to predefined metrics. The analysis is conducted before going into production to identify anomalies. In this paper, they describe a prototypical implementation of the aforementioned idea in the form of a proof-of-concept [10].

[**S38**] This article reports a controlled experiment that compares the effort to implement changes, the correctness and the maintainability of an existing application between two projects; one that uses qualitative dashboards depicting contextual information, and one that does not [17].

[**S39**] In this paper conducts an extensive empirical study to evaluate network embedding algorithms in bug prediction by utilizing and extending node2defect, a newly proposed bug prediction model that combines the embedded vectors with traditional software engineering metrics through concatenation. Experiments are conducted based on seven network embedding algorithms,two effort-aware models, and 13 open-source Java systems [50].

[**S40**] This paper presents a technology for prescriptive software analytics. Their planner offers users a guidance on what action to take in order to improve the quality of a software project. Our preferred planning tool is BELLTREE, which performs cross-project planning with encouraging results.With our BELLTREE planner, we show that it is possible to reduce several hundred defects in software projects [33].

[**S41**] In this paper they investigate whether conclusions in the area of defect prediction, if the claims of the researchers are stable throughout time. This case study

provides evidence that in the field of defect prediction the context of evaluation (in our case, time) plays an important role [5].

[**S42**] In this paper, they propose a two-step approach to first identify whether a commit describes developer-related refactoring events, then to classify it according to the refactoring common quality improvement categories [2].

General Statistics

See Fig. 9, 10, 11, 12, 13 and Table 12, 13, 14, and 15.

Appendix 3: Studies Appraisal

The following acronyms were used for SLR results interpretation (see Table 16):

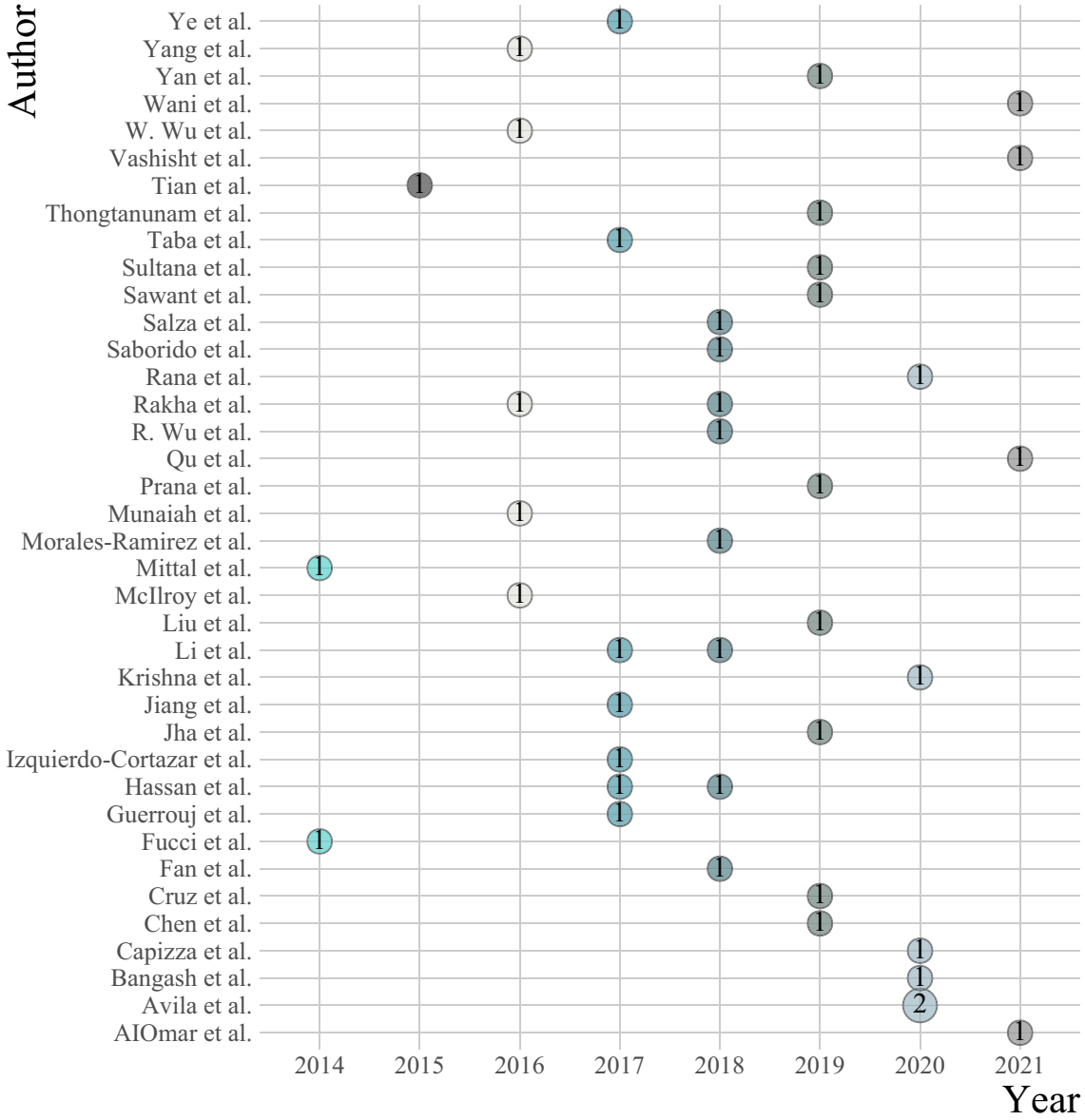

Fig. 9 Number of studies published by each main author over the years

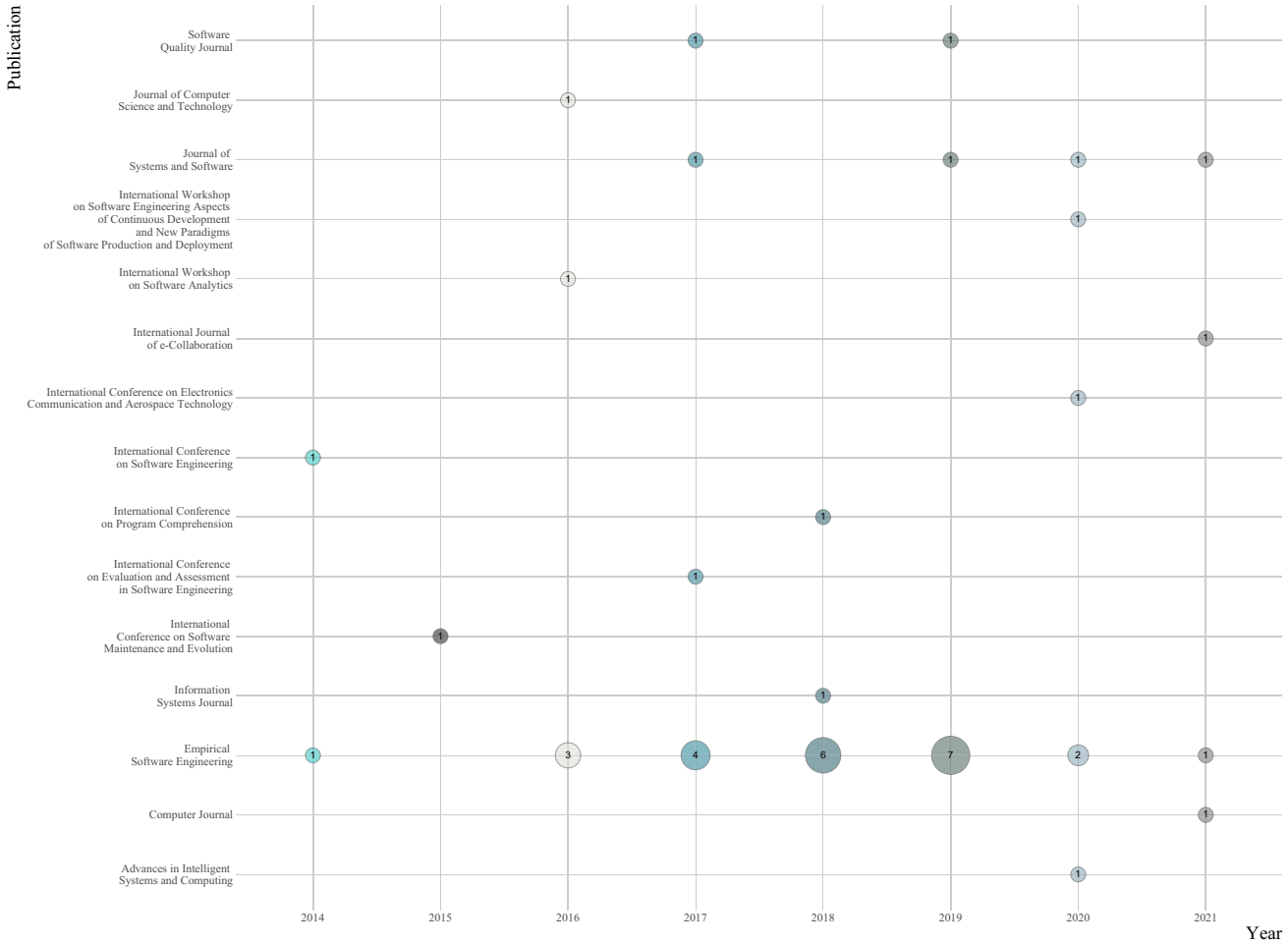

Fig. 10 Frequencies of studies per publisher over the years

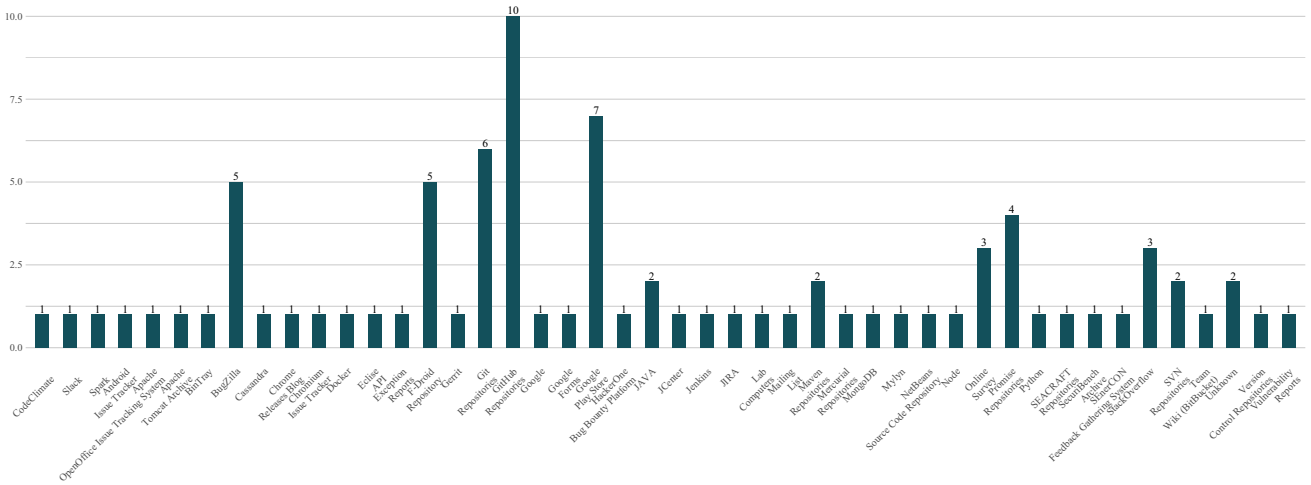

Fig. 11 Frequencies of studies for data sources

Table 12 List of all contributors

Name	Freq.	%	Ref.
Ahmed E. Hassan	10	23.81	[S03], [S08], [S12], [S17], [S19], [S20], [S21], [S25], [S28], [S32]
David Lo	7	16.67	[S06], [S08], [S11], [S24], [S28], [S29], [S31]
Weiyi Shang	5	11.9	[S03], [S19], [S21], [S25], [S32]
Xin Xia	4	9.52	[S08], [S11], [S29], [S31]
Foutse Khomh	3	7.14	[S04], [S13], [S14]
Giuliano Antoniol	3	7.14	[S04], [S13], [S14]
Yann-Gael Guéhéneuc	3	7.14	[S04], [S13], [S14]
Cor-Paul Bezemer	2	4.76	[S17], [S20]
Heng Li	2	4.76	[S19], [S25]
Mohamed Sami Rakha	2	4.76	[S17], [S21]
Safwat Hassan	2	4.76	[S03], [S20]
Shanping Li	2	4.76	[S08], [S11]
Shing-Chi Cheung	2	4.76	[S07], [S10]
Ying Zou	2	4.76	[S05], [S25]
Zhenchang Xing	2	4.76	[S22], [S30]
Abdul Ali Bangash	1	2.38	[S41]
Abram Hindle	1	2.38	[S41]
Ajay Kumar Jha	1	2.38	[S02]
Alberto Bacchelli	1	2.38	[S23]
Ali Ouni	1	2.38	[S42]
Anand Ashok Sawant	1	2.38	[S23]
Andrea De Lucia	1	2.38	[S09]
Andrew Meneely	1	2.38	[S27]
Anna Perini	1	2.38	[S18]
Antonio Capizzi	1	2.38	[S37]
Arianne Navarro Lepe	1	2.38	[S33]
Ashish Sureka	1	2.38	[S16]
Ayrton Mondragon Mejia	1	2.38	[S33]
Benjamin C. M. Fung	1	2.38	[S14]
Bram Adams	1	2.38	[S04]
Burak Turhan	1	2.38	[S15]
Byron J. Williams	1	2.38	[S01]
Chakkrit Tantithamthavorn	1	2.38	[S20]
Chang Xu	1	2.38	[S10]
Christoph Treude	1	2.38	[S06]
Chunyang Chen	1	2.38	[S30]
Cosmo D'Uva	1	2.38	[S09]
Daniel Izquierdo-Cortazar	1	2.38	[S26]
Dario Di Nucci	1	2.38	[S09]
Davide Fucci	1	2.38	[S15]
Deheng Ye	1	2.38	[S22]
Ejaz ul Haq	1	2.38	[S35]
Eklavya Bhatia	1	2.38	[S35]
Eman Abdullah AlOmar	1	2.38	[S42]
Evgeny Bobrov	1	2.38	[S37]
Fabio Palomba	1	2.38	[S09]
Ferdian Thung	1	2.38	[S06]
Filomena Ferrucci	1	2.38	[S09]
Fitsum Meshesha Kifetew	1	2.38	[S18]
Garvit Rana	1	2.38	[S35]

Table 12 (continued)

Name	Freq.	%	Ref.
Gede Artha Azriadi Prana	1	2.38	[S06]
Hareem Sahar	1	2.38	[S41]
Heng Yin	1	2.38	[S39]
Hongyu Zhang	1	2.38	[S07]
Iman Keivanloo	1	2.38	[S05]
Ismael Solis Moreno	1	2.38	[S33]
Itzel Morales-Ramirez	1	2.38	[S18]
Javaid Iqbal Bhat	1	2.38	[S34]
Jesus M. Gonzalez-Barahona	1	2.38	[S26]
Jiahuan He	1	2.38	[S31]
Jian-Ling Sun	1	2.38	[S29]
Jian Zhang	1	2.38	[S10]
Jing Jiang	1	2.38	[S31]
Jorge Luis Victória Barbosa	1	2.38	[S38]
Jue Wang	1	2.38	[S10]
Jun Yan	1	2.38	[S10]
Kaisar Javeed Giri	1	2.38	[S34]
Karim Ali	1	2.38	[S41]
Kazi Zakia Sultana	1	2.38	[S01]
Kleinner Silva Farias de Oliveira	1	2.38	[S38]
Lars Kurth	1	2.38	[S26]
Latifa Guerrouj	1	2.38	[S14]
Leandro Ferreira D'Avila	1	2.38	[S38]
Li Zhang	1	2.38	[S31]
Lili Wei	1	2.38	[S10]
Luis Cruz	1	2.38	[S24]
Luiz J. P. Araújo	1	2.38	[S37]
Manuel Mazzara	1	2.38	[S37]
Megha Mittal	1	2.38	[S16]
Meiyappan Nagappan	1	2.38	[S28]
Meng Yan	1	2.38	[S08]
MingWen	1	2.38	[S07]
Mohamed Wiem Mkaouer	1	2.38	[S42]
Muhammad Ahmad	1	2.38	[S37]
Nachiket Kapre	1	2.38	[S22]
Nasir Ali	1	2.38	[S12]
Nelson Sekitoleko	1	2.38	[S26]
Nuthan Munaiah	1	2.38	[S27]
Pasquale Salza	1	2.38	[S09]
Patanamon Thongtanunam	1	2.38	[S32]
Patricia Ortegon Cano	1	2.38	[S33]
Pavneet Singh Kochhar	1	2.38	[S31]
Rahul Katarya	1	2.38	[S35]
Rahul Krishna	1	2.38	[S40]
Rodrigo Morales	1	2.38	[S13]
Rohit Vashisht	1	2.38	[S36]
Romain Robbes	1	2.38	[S23]
RongxinWu	1	2.38	[S07]
Rubén Saborido	1	2.38	[S13]
Rui Abreu	1	2.38	[S24]
Salvatore Distefano	1	2.38	[S37]

Table 12 (continued)

Name	Freq.	%	Ref.
Seyyed Ehsan Salamati Taba	1	2.38	[S05]
Shaohua Wang	1	2.38	[S05]
Silvana De Gyves Avila	1	2.38	[S33]
Stuart McIlroy	1	2.38	[S12]
Sunghee Lee	1	2.38	[S02]
Syed Afzal Murtaza Rizvi	1	2.38	[S36]
Tanmay Bhowmik	1	2.38	[S01]
Thushari Atapattu	1	2.38	[S06]
Tianyong Wu	1	2.38	[S10]
Tim Menzies	1	2.38	[S40]
Tse-Hsun (Peter) Chen	1	2.38	[S19]
Venera Arnaoudova	1	2.38	[S14]
Wei Wu	1	2.38	[S04]
Woo Jin Lee	1	2.38	[S02]
Xin-Li Yang	1	2.38	[S29]
Yang Liu	1	2.38	[S30]
Yepang Liu	1	2.38	[S10]
Yu QU	1	2.38	[S39]
Yuan Tian	1	2.38	[S28]
Yuanrui Fan	1	2.38	[S11]
Zahid Hussain Wani	1	2.38	[S34]
Zeinab Kermansaravi	1	2.38	[S14]
Zhi-Yuan Wan	1	2.38	[S29]

– **Study Type**

ACM—Analyze and Compare Methodologies, **CS**-Case Study, **CE**-Controlled Experiment
ECS—Exploratory Case Study, **QE**—Quasi-Experiment, **S**—Survey

– **SDLC Activities**

D—Debugging, **I**—Implementation, **M**—Maintenance, **O**—Operations, **T**—Testing

– **Project Stakeholders**

D—Developers, **E**—Educators, **EU**—End-Users, **T**—Testers, **PM**—Product Managers
PjM—Project Managers, **R**—Researchers, **RE**—Requirements Engineers

– **Analytics Scope**

Des—Descriptive Analytics, **Dia**—Diagnostics Analytics
Pred—Predictive Analytics, **Pres**—Prescriptive Analytics

The following taxonomy was used to assess the SDLC contributions:

– **The benefit is:**

Absent (0) ○ Not addressed
Weak (0.25) ◔ Implicitly addressed
Moderate (0.5) ◑ Explicitly addressed (not detailed)
Strong (0.75) ◕ Explained with details and implications
Complete (1) ● Fully explained, validated and replicable

Table 13 Statistics per Institution

Institution	Freq.	%	Ref.
Queen's University	11	26.19	[S03], [S05], [S08], [S12], [S17], [S19], [S20], [S21], [S25], [S28], [S32]
Singapore Management University	7	16.67	[S06], [S08], [S11], [S24], [S28], [S29], [S31]
Concordia University	4	9.52	[S03], [S19], [S25], [S32]
Zhejiang University	4	9.52	[S08], [S11], [S29], [S31]
École Polytechnique de Montréal	3	7.14	[S04], [S13], [S14]
Monash University	3	7.14	[S08], [S11], [S30]
Rochester Institute of Technology	3	7.14	[S27], [S28], [S42]
Hong Kong University of Science and Technology	2	4.76	[S07], [S10]
Nanyang Technological University	2	4.76	[S22], [S30]
University of Adelaide	2	4.76	[S06], [S20]
University of Zurich	2	4.76	[S09], [S23]
Australian National University	1	2.38	[S30]
Beihang University	1	2.38	[S31]
Bitergia	1	2.38	[S26]
Citrix	1	2.38	[S26]
Columbia University	1	2.38	[S40]
Delft University of Technology	1	2.38	[S23]
Delhi Technological University	1	2.38	[S35]
École de Technologie Supérieure	1	2.38	[S14]
ETS Montreal, University of Quebec	1	2.38	[S42]
Fondazione Bruno Kessler	1	2.38	[S18]
Free University of Bozen-Bolzano	1	2.38	[S23]
IBM	1	2.38	[S33]
Indraprastha Institute of Information Technology	1	2.38	[S16]
INESC ID	1	2.38	[S24]
INFOTEC	1	2.38	[S18]
Innopolis University	1	2.38	[S37]
Islamic University of Science and Technology	1	2.38	[S34]
Jamia Millia Islamia	1	2.38	[S36]
Kyungpook National University	1	2.38	[S02]
McGill University	1	2.38	[S14]
Mississippi State University	1	2.38	[S01]
Nanjing University	1	2.38	[S10]
NC State University	1	2.38	[S40]
Southern University of Science and Technology	1	2.38	[S10]
Universidad Rey Juan Carlos	1	2.38	[S26]
Universitá della Svizzera Italiana	1	2.38	[S09]
University of Alberta	1	2.38	[S41]
University of California	1	2.38	[S39]
University of Chinese Academy of Sciences	1	2.38	[S10]
University of Lisbon	1	2.38	[S24]
University of Melbourne	1	2.38	[S32]
University of Messina	1	2.38	[S37]
University of Newcastle	1	2.38	[S07]
University of Oulu	1	2.38	[S15]
University of Salerno	1	2.38	[S09]
University of Vale do Rio dos Sinos	1	2.38	[S38]
University of Waterloo	1	2.38	[S12]
Vrije Universiteit Brussel	1	2.38	[S09]

Table 13 (continued)

Institution	Freq.	%	Ref.
Washington State University	1	2.38	[S14]

Table 14 Statistics per continent and country

	Freq.	%	Ref.
Continent			
North America	23	54.76	[S01], [S03], [S04], [S05], [S08], [S12], [S13], [S14], [S17], [S18], [S19], [S20], [S21], [S25], [S26], [S27], [S28], [S32], [S33], [S39], [S40], [S41], [S42]
Asia	17	40.48	[S02], [S06], [S07], [S08], [S10], [S11], [S16], [S22], [S24], [S28], [S29], [S30], [S31], [S34], [S35], [S36], [S37]
Europe	7	16.67	[S09], [S15], [S18], [S23], [S24], [S26], [S37]
Oceania	7	16.67	[S06], [S07], [S08], [S11], [S20], [S30], [S32]
South America	1	2.38	[S38]
Country			
Canada	16	38.1	[S03], [S04], [S05], [S08], [S12], [S13], [S14], [S17], [S19], [S20], [S21], [S25], [S28], [S32], [S41], [S42]
Singapore	9	21.43	[S06], [S08], [S11], [S22], [S24], [S28], [S29], [S30], [S31]
USA	8	19.05	[S01], [S14], [S26], [S27], [S28], [S39], [S40], [S42]
Australia	7	16.67	[S06], [S07], [S08], [S11], [S20], [S30], [S32]
China	6	14.29	[S07], [S08], [S10], [S11], [S29], [S31]
India	4	9.52	[S16], [S34], [S35], [S36]
Italy	4	9.52	[S09], [S18], [S23], [S37]
Mexico	2	4.76	[S18], [S33]
Switzerland	2	4.76	[S09], [S23]
Belgium	1	2.38	[S09]
Brazil	1	2.38	[S38]
Finland	1	2.38	[S15]
Portugal	1	2.38	[S24]
Republic of Korea	1	2.38	[S02]
Russian	1	2.38	[S37]
Spain	1	2.38	[S26]
The Netherlands	1	2.38	[S23]

Table 15 Data sources findings (frequency > 1)

Data Sources	Freq.	%	Ref.
GitHub Repositories	10	23.81	[S02], [S06], [S09], [S10], [S13], [S23], [S24], [S31], [S37], [S42]
Google Play Store	7	16.67	[S03], [S05], [S10], [S12], [S20], [S24], [S28]
Git Repositories	6	14.29	[S08], [S14], [S16], [S19], [S26], [S32]
BugZilla	5	11.9	[S07], [S14], [S16], [S17], [S21]
F-Droid Repository	5	11.9	[S02], [S03], [S09], [S10], [S24]
Promise Repositories	4	9.52	[S35], [S39], [S40], [S41]
Online Survey	3	7.14	[S15], [S23], [S24]
StackOverflow	3	7.14	[S22], [S29], [S30]
JAVA	2	4.76	[S35], [S38]
Maven Repositories	2	4.76	[S04], [S09]
SVN Repositories	2	4.76	[S09], [S14]
Unknown	2	4.76	[S34], [S36]
Android Issue Tracker	1	2.38	[S27]
Apache OpenOffice Issue Tracking System	1	2.38	[S18]
Apache Tomcat Archive	1	2.38	[S01]
BinTray	1	2.38	[S09]
Cassandra	1	2.38	[S33]
Chrome Releases Blog	1	2.38	[S27]
Chromium Issue Tracker	1	2.38	[S27]
CodeClimate	1	2.38	[S37]
Docker	1	2.38	[S37]
Eclise API	1	2.38	[S38]
Exception Reports	1	2.38	[S07]
Gerrit	1	2.38	[S11]
Google	1	2.38	[S09]
Google Forms	1	2.38	[S13]
HackerOne Bug Bounty Platform	1	2.38	[S27]
JCenter	1	2.38	[S09]
Jenkins	1	2.38	[S37]
JIRA	1	2.38	[S14]
Lab Computers	1	2.38	[S15]
Mailing List	1	2.38	[S26]
Mercurial Repositories	1	2.38	[S16]
MongoDB	1	2.38	[S33]
Mylyn	1	2.38	[S38]
NetBeans Source Code Repository	1	2.38	[S07]
Node	1	2.38	[S37]
Python	1	2.38	[S33]
SEACRAFT Repositories	1	2.38	[S35]
SecuriBench Archive	1	2.38	[S01]
SEnerCON Feedback Gathering System	1	2.38	[S18]
Slack	1	2.38	[S37]
Spark	1	2.38	[S33]
Team Wiki (BitBucket)	1	2.38	[S16]
Version Control Repositories	1	2.38	[S25]
Vulnerability Reports	1	2.38	[S01]

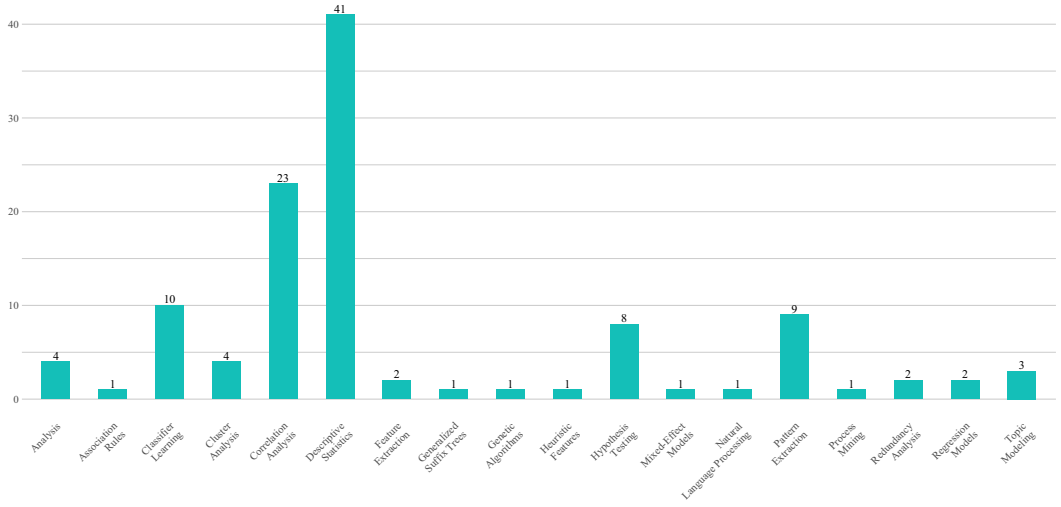

Fig. 12 Frequencies of studies for mining methods

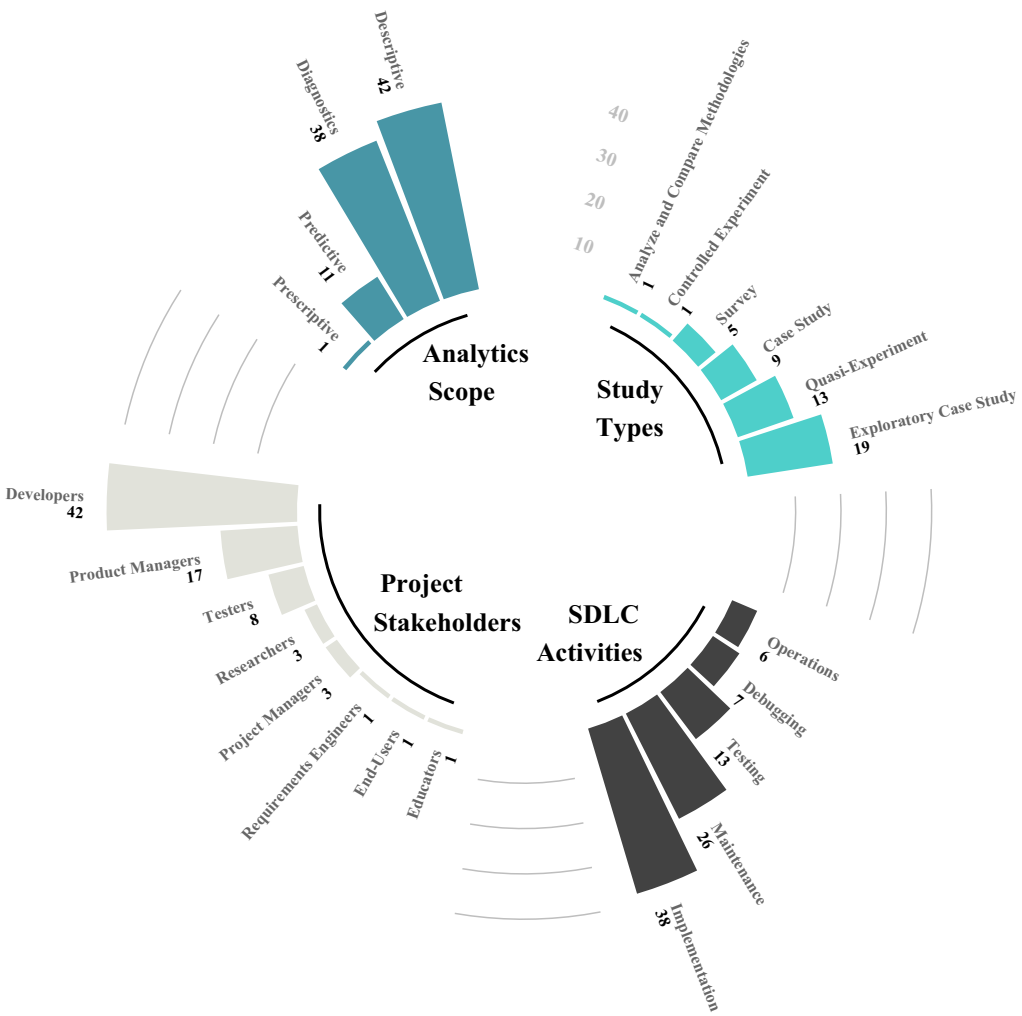

Fig. 13 Frequencies of studies combining multiple RQs in the SLR

Table 16 Systematic literature review results

Study	Study Type	Data Sources	Process Perspective	SDLC Activities	Project Stakeholders	Mining Methods	Analytics Scope	Contributions to SDLC				
								Technical Debt	Time Management	Costs Control	Risks Assessment	Security Analysis
S01	CS	Vulnerability Reports,Apache Tomcat Archive,SecuriBench Archive	Post-Mortem	I,T	D,T	Descriptive Statistics,Pattern Extraction,Correlation Analysis	Des,Dia	◕	◔	◔	◔	◕
S02	CS	F-Droid Repository,GitHub Repositories	Post-Mortem	I	D	Descriptive Statistics,Pattern Extraction,Correlation Analysis	Des,Dia	◕	◔	◔	○	○
S03	CS	F-Droid Repository,Google Play Store	Post-Mortem	O	D,PM	Descriptive Statistics,Pattern Extraction	Des,Dia	◑	○	○	○	○
S04	ECS	Maven Repositories	Post-Mortem	I	D	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis	Des,Dia	◕	○	◔	○	○
S05	ECS	Google Play Store	Post-Mortem	O	D	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis	Des,Dia	◕	○	○	○	○
S06	QE,S	GitHub Repositories	Post-Mortem	I	D,PM	Descriptive Statistics,Pattern Extraction,Classifier Learning	Des,Dia,Pred	◕	○	○	○	○
S07	QE	NetBeans Source Code Repository,BugZilla,Exception Reports	Post-Mortem	I,D,M	D	Descriptive Statistics,Pattern Extraction,Heuristic Features,Classifier Learning	Des,Dia	◕	○	○	○	○
S08	CS	Git Repositories	Post-Mortem	I,D,M	D	Descriptive Statistics,Feature Extraction,Correlation Analysis,Redundancy Analysis,Classifier Learning	Des,Dia,Pred	◕	◔	◔	○	○
S09	ECS	F-Droid Repository,SVN Repositories,GitHub Repositories,BinTray,JCenter,Maven Repositories,Google	Post-Mortem	I,D,M	D	Descriptive Statistics,Pattern Extraction	Des,Dia	◕	○	○	○	○
S10	ECS	F-Droid Repository,GitHub Repositories,Google Play Store	Post-Mortem	I,D,M	D	Descriptive Statistics,Pattern Extraction	Des,Dia	◕	○	○	○	○
S11	QE	Gerrit	Post-Mortem	I,D,M	D	Descriptive Statistics,Hyphotesis Testing,Redundancy Analysis,Feature Extraction,Correlation Analysis,Classifier Learning	Des,Dia,Pred	◕	◔	◔	○	○
S12	QE	Google Play Store	Post-Mortem	I,D,M	D,PM	Descriptive Statistics	Des,Dia	◕	○	○	○	○
S13	ECS,QE,S	GitHub Repositories,Google Forms	Post-Mortem	I,M	D	Descriptive Statistics,Pattern Extraction	Des,Dia	◕	○	○	○	○
S14	QE	Git Repositories,SVN Repositories,BugZilla,JIRA	Post-Mortem	I,M	D	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis	Des,Dia	◕	○	○	○	○
S15	QE	Online Survey,Lab Computers	Post-Mortem	I	D	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis	Des,Dia	◕	◑	○	○	○
S16	ECS	Team Wiki (BitBucket),Mercurial Repositories,Git Repositories,BugZilla	Post-Mortem	I	D	Descriptive Statistics,Process Mining	Des,Dia	◕	○	○	○	○
S17	QE	BugZilla	Post-Mortem	I,M	D,R	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis	Des,Dia	◕	○	○	○	○
S18	QE	Apache OpenOffice Issue Tracking System,SEnerCON Feedback Gathering System	Post-Mortem	I,M,O	D,PM,RE	Descriptive Statistics,Hyphotesis Testing,Correlation Analysis,Classifier Learning	Des,Dia,Pred	◕	◔	○	○	○
S19	QE	Git Repositories	Post-Mortem	I	D	Descriptive Statistics,Correlation Analysis,Topic Modeling,Regression Models	Des,Dia,Pred	◕	◔	○	○	○
S20	ECS	Google Play Store	Post-Mortem	I,M,O	D,EU,PM,R	Descriptive Statistics,Correlation Analysis,Mixed-Effect Models,Cluster Analysis,Regression Models	Des,Dia,Pred	◕	○	○	○	○

Table 16 (continued)

S21	QE	BugZilla	Post-Mortem	I,M	D	Descriptive Statistics,Correlation Analysis,Classifier Learning	Des,Dia,Pred	◔	◑	◔	○	○
S22	ECS	StackOverflow	Post-Mortem	I,M	D	Descriptive Statistics,Correlation Analysis,Topic Modeling,Cluster Analysis	Des,Dia	◑	○	○	○	○
S23	ECS,S	GitHub Repositories,Online Survey	Post-Mortem	I,M	D	Descriptive Statistics,Pattern Extraction	Des,Dia	◑	◔	○	○	○
S24	QE,S	F-Droid Repository,GitHub Repositories,Google Play Store,Online Survey	Post-Mortem	I,M,T	D,T	Descriptive Statistics,Correlation Analysis	Des,Dia	◕	○	○	○	○
S25	ECS	Version Control Repositories	Post-Mortem	I,M	D	Descriptive Statistics,Correlation Analysis,Classifier Learning,Cluster Analysis	Des,Dia,Pred	◕	○	○	○	○
S26	ECS	Mailing List,Git Repositories	Post-Mortem	I,M	D,PjM	Descriptive Statistics	Des,Dia	◑	◑	○	○	○
S27	ECS	Android Issue Tracker,Chrome Releases Blog,Chromium Issue Tracker,HackerOne Bug Bounty Platform	Post-Mortem	I,M	D,PM	Descriptive Statistics,Correlation Analysis	Des,Dia	◔	○	○	○	◑
S28	ECS	Google Play Store	Post-Mortem	I,M,O	D,PM	Descriptive Statistics,Correlation Analysis	Des,Dia	◑	○	○	○	○
S29	ECS	StackOverflow	Post-Mortem	I,M	D,R,PjM,E	Descriptive Statistics,Topic Modeling,Genetic Algorithms	Des,Dia	◔	○	○	○	◑
S30	ECS	StackOverflow	Post-Mortem	I,M,T	D	Descriptive Statistics,Association Rules,Natural Language Processing	Des,Dia,Pres	◕	◔	○	○	◔
S31	ECS,S	GitHub Repositories	Post-Mortem	I	D	Descriptive Statistics	Des,Dia	◑	○	○	○	○
S32	ECS	Git Repositories	Post-Mortem	I	D	Descriptive Statistics,Generalized Suffix Trees,Correlation Analysis,Cluster Analysis,Classifier Learning	Des,Dia,Pred	◕	○	○	○	○
S33	CS	MongoDB,Python, Spark,Cassandra	Post-Mortem	I,T	D,T,PM	Descriptive Statistics,Analysis	Des	◔	○	○	○	○
S34	QE	Unknown	Post-Mortem	I,M,T	D,T,PM	Descriptive Statistics,Analysis	Des,Dia	◔	◑	◑	○	○
S35	ACM	Promise Repositories,JAVA,SEACRAFT Repositories	Pre-Mortem	I,M,O,T	D,T,PM	Analysis	Des	◑	◑	◑	◔	○
S36	CS	Unknown	Post-Mortem	I,T	D,PM	Descriptive Statistics	Des,Dia	◕	◑	◑	◔	◔
S37	ECS	GitHub Repositories,Jenkins, CodeClimate, Docker, Slack,Node	Post-Mortem	I,T	D,T,PM,PjM	Descriptive Statistics	Des	◕	◑	◑	◔	○
S38	CE	JAVA,Eclise API,Mylyn	Post-Mortem	M,T	D,PM	Descriptive Statistics,Analysis	Des,Dia	◑	◑	◔	○	○
S39	CS	Promise Repositories	Post-Mortem	D,M,T	D,PM	Descriptive Statistics,Hypothesis Testing,Correlation Analysis	Des,Dia,Pred	◕	◑	◔	○	○
S40	ECS	Promise Repositories	Post-Mortem	I,M,T	D,T,PM	Descriptive Statistics,Correlation Analysis,Classifier Learning	Des,Dia,Pred	◕	◑	◔	○	○
S41	CS	Promise Repositories	Post-Mortem	I,T	D,PM	Descriptive Statistics,Correlation Analysis,Classifier Learning	Des,Dia	◕	◑	○	○	○
S42	CS	GitHub Repositories	Post-Mortem	I,T	D,T,PM	Descriptive Statistics,Correlation Analysis	Des	◑	○	○	○	○

Acknowledgements This work was partially funded by the Portuguese Foundation for Science and Technology, under ISTAR's projects UIDB/04466/2020 and UIDP/04466/2020.

References

1. Abdellatif M, Capretz F, Ho D (2015) Software Analytics to software practice: a systematic literature review. In: 1st International workshop on big data software engineering, IEEE/ACM, New York, pp 30–36. https://doi.org/10.1109/BIGDSE.2015.14. https://www.eng.uwo.ca/Electrical/faculty/capretz_l/docs/publications/Tamer-BIGDSE-v2.pdf
2. AlOmar EA, Mkaouer MW, Ouni A (2021) Toward the automatic classification of self-affirmed refactoring. J Syst Softw 171:110821. https://doi.org/10.1016/J.JSS.2020.110821
3. Anwar H, Pfahl D (2017) Towards greener software engineering using software analytics: a systematic mapping. In: Proceedings of 43rd Euromicro conference on software engineering and advanced applications, SEAA 2017. Institute of Electrical and Electronics

Engineers Inc., pp 157–166. https://doi.org/10.1109/SEAA.2017.56
4. Avila SDG, Cano PO, Mejia AM, Moreno IS, Lepe AN (2020) A data driven platform for improving performance assessment of software defined storage solutions. Adv Intell Syst Comput 1071:266–275. https://doi.org/10.1007/978-3-030-33547-2_20
5. Bangash AA, Sahar H, Hindle A, Ali K (2020) On the time-based conclusion stability of cross-project defect prediction models. Empir Softw Eng 25:5047–5083. https://doi.org/10.1007/S10664-020-09878-9
6. Buse RPL, Zimmermann T (2010) Analytics for software development. Tech. rep., Microsoft Research. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-111.pdf
7. Buse RP, Zimmermann T (2012) Information needs for software development analytics. In: Proceedings - International Conference on Software Engineering, pp 987–996, https://doi.org/10.1109/ICSE.2012.6227122
8. Cai KY (2002) Optimal software testing and adaptive software testing in the context of software cybernetics. Inf Softw Technol 44(14):841–855. https://doi.org/10.1016/S0950-5849(02)00108-8
9. Cai KY, Chen T, Tse T (2002) Towards research on software cybernetics. In: 7th IEEE international symposium on high assurance systems engineering, 2002. Proceedings, pp 240–241. https://doi.org/10.1109/HASE.2002.1173129
10. Capizzi A, Distefano S, Araújo LJ, Mazzara M, Ahmad M, Bobrov E (2020) Anomaly detection in devops toolchain. Lecture notes in computer science (including subseries Lecture notes in artificial intelligence and Lecture notes in bioinformatics), vol 12055, pp 37–51. https://doi.org/10.1007/978-3-030-39306-9_3
11. Chen L, Babar MA (2011) A systematic review of evaluation of variability management approaches in software product lines. Inf Softw Technol 53(4):344–362
12. Chen C, Xing Z, Liu Y (2019) What's Spain's Paris? Mining analogical libraries from Q & A discussions. Empir Softw Eng 24(3):1155–1194. https://doi.org/10.1007/s10664-018-9657-y
13. Cosentino V, Izquierdo JL, Cabot J (2017) A systematic mapping study of software development with GitHub. IEEE Access 5:7173–7192. https://doi.org/10.1109/ACCESS.2017.2682323
14. Cruz L, Abreu R, Lo D (2019) To the attention of mobile software developers: guess what, test your app! Empir Softw Eng 24:2438–2468. https://doi.org/10.1007/s10664-019-09701-0
15. Dasanayake S, Markkula J, Oivo M (2014) Concerns in software development: a systematic mapping study. In: Proceedings of the 18th International conference on evaluation and assessment in software engineering. Association for Computing Machinery, pp 1–4. https://doi.org/10.1145/2601248.2601290
16. Davenport TH, Harris JG, Morison R (2010) Analytics at work: smarter decisions, better results. Harvard Business Press. http://discovery.uoc.edu/iii/encore/record/C__Rb1049687__SAnalytics%20at%20Work__Orightresult__U__X7?lang=spi
17. D'Avila LF, Farias K, Barbosa JLV (2020) Effects of contextual information on maintenance effort: a controlled experiment. J Syst Softw. https://doi.org/10.1016/J.JSS.2019.110443
18. Dybå T, Dingsøyr T (2008) Strength of evidence in systematic reviews in software engineering. In: ESEM'08: proceedings of the 2008 ACM-IEEE international symposium on empirical software engineering and measurement, pp 178–187. https://doi.org/10.1145/1414004.1414034
19. Emam KE, Koru AG (2008) A replicated survey of IT software project failures. IEEE Softw 25(5):84–90. https://doi.org/10.1109/MS.2008.107. (**ieeexplore.ieee.org/document/4602680/**)
20. Fan Y, Xia X, Lo D, Li S (2018) Early prediction of merged code changes to prioritize reviewing tasks. Empir Softw Eng 23(6):3346–3393. https://doi.org/10.1007/s10664-018-9602-0
21. Fucci D, Turhan B (2014) On the role of tests in test-driven development: a differentiated and partial replication. Empir Softw Eng 19(2):277–302. https://doi.org/10.1007/s10664-013-9259-7
22. Garcia CdS, Meincheim A, Faria Junior ER, Dallagassa MR, Sato DMV, Carvalho DR, Santos EAP, Scalabrin EE (2019) Process mining techniques and applications—a systematic mapping study. Expert Syst Appl 133:260–295. https://doi.org/10.1016/j.eswa.2019.05.003
23. Gomes TL, Oliveira TC, Cowan D, Alencar P (2014) Mining reuse processes. In: CIBSE 2014: proceedings of the 17th Ibero-American conference software engineering. Curran Associates, Pucon, pp 179–191. https://dblp.org/rec/bib/conf/cibse/GomesOCA14
24. Guerrouj L, Kermansaravi Z, Arnaoudova V, Fung BC, Khomh F, Antoniol G, Guéhéneuc YG (2017) Investigating the relation between lexical smells and change- and fault-proneness: an empirical study. Softw Qual J 25(3):641–670. https://doi.org/10.1007/s11219-016-9318-6
25. Hassan S, Shang W, Hassan AE (2017) An empirical study of emergency updates for top android mobile apps. Empir Softw Eng 22(1):505–546. https://doi.org/10.1007/s10664-016-9435-7
26. Hassan S, Tantithamthavorn C, Bezemer CP, Hassan AE (2018) Studying the dialogue between users and developers of free apps in the Google Play Store. Empir Softw Eng 23(3):1275–1312. https://doi.org/10.1007/s10664-017-9538-9
27. IEEE Computer Society (2014) SWEBOK V3.0. No. V3.0 in 1. IEEE Computer Society. https://doi.org/10.1234/12345678, http://www4.ncsu.edu/~tjmenzie/cs510/pdf/SWEBOKv3.pdf
28. Izquierdo-Cortazar D, Sekitoleko N, Gonzalez-Barahona JM, Kurth L (2017) Using metrics to track code review performance. In: ACM international conference proceeding series. Association for Computing Machinery, vol Part F128635, pp 214–223. https://doi.org/10.1145/3084226.3084247
29. Jha AK, Lee S, Lee WJ (2019) An empirical study of configuration changes and adoption in Android apps. J Syst Softw 156:164–180. https://doi.org/10.1016/j.jss.2019.06.095
30. Jiang J, Lo D, He J, Xia X, Kochhar PS, Zhang L (2017) Why and how developers fork what from whom in GitHub. Empirical Softw Eng 22(1):547–578. https://doi.org/10.1007/s10664-016-9436-6
31. Kitchenham B, Brereton P (2013) A systematic review of systematic review process research in software engineering. Inf Softw Technol 55(12):2049–2075. https://doi.org/10.1016/j.infsof.2013.07.010
32. Kitchenham B, Pearl Brereton O, Budgen D, Turner M, Bailey J, Linkman S (2009) Systematic literature reviews in software engineering—a systematic literature review. Inf Softw Technol 5:7–15
33. Krishna R, Menzies T (2020) Learning actionable analytics from multiple software projects. Empir Softw Eng 25:3468–3500. https://doi.org/10.1007/S10664-020-09843-6
34. Li H, Shang W, Zou Y, Hassan E, A, (2017) Towards just-in-time suggestions for log changes. Empir Softw Eng 22(4):1831–1865. https://doi.org/10.1007/s10664-016-9467-z
35. Li H, Chen THP, Shang W, Hassan AE (2018) Studying software logging using topic models. Empir Softw Eng 23(5):2655–2694. https://doi.org/10.1007/s10664-018-9595-8
36. Liu Y, Wang J, Wei L, Xu C, Cheung SC, Wu T, Yan J, Zhang J (2019) DroidLeaks: a comprehensive database of resource leaks in Android apps. Empir Softw Eng 24(6):3435–3483. https://doi.org/10.1007/s10664-019-09715-8
37. McIlroy S, Ali N, Hassan AE (2016) Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store. Empir Softw Eng 21(3):1346–1370. https://doi.org/10.1007/s10664-015-9388-2
38. Menzies T, Bird C, Zimmermann T, Schulte W, Kocaganeli E (2011) The inductive software engineering manifesto: principles

for industrial data mining. In: Proceedings of the international workshop on machine learning technologies in software engineering. Association for Computing Machinery, pp 19–26. http://bit.ly/o02QZJ
39. Menzies T, Minku L, Peters F (2015) The art and science of analyzing software data; quantitative methods. In: Proceedings of the international conference on software engineering, vol 2. IEEE Computer Society, pp 959–960. https://doi.org/10.1109/ICSE.2015.306
40. Mittal M, Sureka A (2014a) MIMANSA: process mining software repositories from student projects in an undergraduate software engineering course categories and subject descriptors. Softw Eng Educ Train ICSE 2014:344–353
41. Mittal M, Sureka A (2014b) Process mining software repositories from student projects in an undergraduate software engineering course. In: 36th International conference on software engineering, ICSE Companion 2014—proceedings. Association for Computing Machinery, pp 344–353. https://doi.org/10.1145/2591062.2591152
42. Mohagheghi P, Conradi R (2007) Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empir Softw Eng 12(5):471–516. https://doi.org/10.1007/s10664-007-9040-x
43. Mohagheghi P, Jorgensen M (2017) What contributes to the success of IT projects? Success factors, challenges and lessons learned from an empirical study of software projects in the Norwegian public sector. In: 2017 IEEE/ACM 39th international conference on software engineering companion (ICSE-C). IEEE, pp 371–373. https://doi.org/10.1109/ICSE-C.2017.146, http://ieeexplore.ieee.org/document/7965362/
44. Morales-Ramirez I, Kifetew FM, Perini A (2018) Speech-acts based analysis for requirements discovery from online discussions. Inf Syst 86:94–112. https://doi.org/10.1016/j.is.2018.08.003
45. Munaiah N, Meneely A (2016) Vulnerability severity scoring and bounties: why the disconnect. In: SWAN 2016 - Proceedings of the 2nd international workshop on software analytics, co-located with FSE 2016. Association for Computing Machinery, pp 8–14. https://doi.org/10.1145/2989238.2989239
46. Nakamoto S (2009) Bitcoin: A Peer-to-Peer Electronic Cash System. Tech. rep., http://www.bitcoin.org, www.bitcoin.org
47. Nayebi M, Ruhe G, Mota RC, Mufti M (2016) Analytics for software project management—wWhere are we and where do we go? In: Proceedings—2015 30th IEEE/ACM international conference on automated software engineering workshops, ASEW 2015. Institute of Electrical and Electronics Engineers, pp 18–21. https://doi.org/10.1109/ASEW.2015.28
48. Poncin W, Serebrenik A, Brand MVD (2011) Process mining software repositories. In: 2011 15th European conference on software maintenance and reengineering, pp 5–14. https://doi.org/10.1109/CSMR.2011.5
49. Prana GAA, Treude C, Thung F, Atapattu T, Lo D (2019) Categorizing the content of GitHub README files. Empir Softw Eng 24(3):1296–1327. https://doi.org/10.1007/s10664-018-9660-3
50. Qu Y, Yin H (2021) Evaluating network embedding techniques' performances in software bug prediction. Empir Softw Eng. https://doi.org/10.1007/S10664-021-09965-5
51. Rakha MS, Shang W, Hassan AE (2016) Studying the needed effort for identifying duplicate issues. Empir Softw Eng 21(5):1960–1989. https://doi.org/10.1007/s10664-015-9404-6
52. Rakha MS, Bezemer CP, Hassan AE (2018) Revisiting the performance of automated approaches for the retrieval of duplicate reports in issue tracking systems that perform just-in-time duplicate retrieval. Empir Softw Eng 23(5):2597–2621. https://doi.org/10.1007/s10664-017-9590-5
53. Rana G, Haq EU, Bhatia E, Katarya R (2020) A study of hyperparameter tuning in the field of software analytics. In: Proceedings of the 4th international conference on electronics, communication and aerospace technology, ICECA 2020, pp 455–459. https://doi.org/10.1109/ICECA49313.2020.9297613
54. Rodriguez D, Herraiz I, Harrison R (2012) On software engineering repositories and their open problems. In: 2012 1st International workshop on realizing AI synergies in software engineering, RAISE 2012—pProceedings, pp 52–56. https://doi.org/10.1109/RAISE.2012.6227971
55. Saborido R, Morales R, Khomh F, Guéhéneuc YG, Antoniol G (2018) Getting the most from map data structures in Android. Empir Softw Eng 23(5):2829–2864. https://doi.org/10.1007/s10664-018-9607-8
56. Salza P, Palomba F, Nucci DD, D'uva C, De Lucia A, Ferrucci F (2018) Do developers update third-party libraries in mobile apps. In: Proceedings of the 26th conference on program comprehension, vol 12. Association for Computing Machinery, pp 255–265
57. Sawant AA, Robbes R, Bacchelli A (2019) To react, or not to react: patterns of reaction to API deprecation. Empir Softw Eng 24(6):3824–3870. https://doi.org/10.1007/s10664-019-09713-w
58. Sultana KZ, Williams BJ, Bhowmik T (2019) A study examining relationships between micro patterns and security vulnerabilities. Softw Qual J 27(1):5–41. https://doi.org/10.1007/s11219-017-9397-z
59. Taba SES, Keivanloo I, Zou Y, Wang S (2017) An exploratory study on the usage of common interface elements in android applications. J Syst Softw 131:491–504. https://doi.org/10.1016/j.jss.2016.07.010
60. Tapscott D, Tapscott A (2016) Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world. Portfolio
61. Thongtanunam P, Shang W, Hassan AE (2019) Will this clone be short-lived? Towards a better understanding of the characteristics of short-lived clones. Empir Softw Eng 24(2):937–972. https://doi.org/10.1007/s10664-018-9645-2
62. Tian Y, Nagappan M, Lo D, Hassan AE (2015) What are the characteristics of high-rated apps? A case study on free Android Applications. In: 2015 IEEE 31st International conference on software maintenance and evolution, ICSME 2015—proceedings. Institute of Electrical and Electronics Engineers, pp 301–310. https://doi.org/10.1109/ICSM.2015.7332476
63. Tim Menzies LW, Zimmermann T (2016) Perspectives on data science for software engineering. Elsevier, Amsterdam. https://doi.org/10.1016/C2015-0-00521-4
64. Van Der Aalst W (2016) Process mining: data science in action, 2nd edn. Springer, Berlin. https://doi.org/10.1007/978-3-662-49851-4
65. Van Der Aalst W, Adriansyah A, De Medeiros AKA, Arcieri F, Baier T, Blickle T, Bose JC, Van Den Brand P, Brandtjen R, Buijs J, Burattin A, Carmona J, Castellanos M, Claes J, Cook J, Costantini N, Curbera F, Damiani E, De Leoni M, Delias P, Van Dongen BF, Dumas M, Dustdar S, Fahland D, Ferreira DR, Gaaloul W, Van Geffen F, Goel S, Günther C, Guzzo A, Harmon P, Ter Hofstede A, Hoogland J, Ingvaldsen JE, Kato K, Kuhn R, Kumar A, La Rosa M, Maggi F, Malerba D, Mans RS, Manuel A, McCreesh M, Mello P, Mendling J, Montali M, Motahari-Nezhad HR, Zur Muehlen M, Munoz-Gama J, Pontieri L, Ribeiro J, Rozinat A, Seguel Pérez H, Seguel Pérez R, Sepúlveda M, Sinur J, Soffer P, Song M, Sperduti A, Stilo G, Stoel C, Swenson K, Talamo M, Tan W, Turner C, Vanthienen J, Varvaressos G, Verbeek E, Verdonk M, Vigo R, Wang J, Weber B, Weidlich M, Weijters T, Wen L, Westergaard M, Wynn M (2012) Process mining manifesto. Lecture notes in business information processing 99 (LNBIP), pp 169–194. https://doi.org/10.1007/978-3-642-28108-2_19

66. Vashisht R, Rizvi SAM (2021) An empirical study of heterogeneous cross-project defect prediction using various statistical techniques. Int J e-Collaboration 17:55–71. https://doi.org/10.4018/IJEC.2021040104
67. Wani ZH, Bhat JI, Giri KJ (2021) A generic analogy-centered software cost estimation based on differential evolution exploration process. Comput J 64:462–472. https://doi.org/10.1093/COMJNL/BXAA199
68. Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering (EASE '14), pp 1–10. https://doi.org/10.1145/2601248.2601268
69. Wu R, Wen M, Cheung SC, Zhang H (2018) ChangeLocator: locate crash-inducing changes based on crash reports. Empir Softw Eng 23(5):2866–2900. https://doi.org/10.1007/s10664-017-9567-4
70. Wu W, Khomh F, Adams B, Guéhéneuc YG, Antoniol G (2016) An exploratory study of api changes and usages based on apache and eclipse ecosystems. Empir Softw Eng 21(6):2366–2412. https://doi.org/10.1007/s10664-015-9411-7
71. Yan M, Xia X, Lo D, Hassan AE, Li S (2019) Characterizing and identifying reverted commits. Empir Softw Eng 24(4):2171–2208. https://doi.org/10.1007/s10664-019-09688-8
72. Yang XL, Lo D, Xia X, Wan ZY, Sun JL (2016) What security questions do developers ask? A large-scale study of stack overflow posts. J Comput Sci Technol 31(5):910–924. https://doi.org/10.1007/s11390-016-1672-0. (**archive.org/details/stackexchange**)
73. Yang H, Chen F, Aliyu S (2017) Modern software cybernetics: new trends. J Syst Softw 124:169–186. https://doi.org/10.1016/j.jss.2016.08.095
74. Ye D, Xing Z, Kapre N (2017) The structure and dynamics of knowledge network in domain-specific Q &A sites: a case study of stack overflow. Empir Softw Eng 22(1):375–406. https://doi.org/10.1007/s10664-016-9430-z
75. Zannier C, Melnik G, Maurer F (2006) On the success of empirical studies in the international conference on software engineering. In: Proceedings of international conference on software engineering, pp 341–350. https://doi.org/10.1145/1134285.1134333
76. Zhang D, Han S, Dang Y, Lou JG, Zhang H, Research Asia M, Xie T (2013a) Software analytics in practice. IEEE Softw. http://channel9.msdn
77. Zhang D, Han S, Dang Y, Lou JG, Zhang H, Xie T (2013b) Software analytics in practice. IEEE Softw 30(5):30–37. https://doi.org/10.1109/MS.2013.94
78. Zhang L, Tian JH, Jiang J, Liu YJ, Pu MY, Yue T (2018) Empirical research in software engineering—a literature survey. J Comput Sci Technol 33(5):876–899. https://doi.org/10.1007/s11390-018-1864-x