The Journal of Systems & Software 174 (2021) 110892

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

Change impact analysis: A systematic mapping study☆

Maria Kretsou [a], Elvira-Maria Arvanitou [b], Apostolos Ampatzoglou [b,*], Ignatios Deligiannis [c], Vassilis C. Gerogiannis [d]

[a] Department of Informatics, Open Hellenic University, Patras, Greece
[b] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
[c] Department of Information & Electronic Engineering, International Hellenic University, Greece
[d] Department of Digital Systems, School of Technology, University of Thessaly, Greece

ARTICLE INFO

Article history:
Received 7 July 2020
Received in revised form 28 October 2020
Accepted 22 December 2020
Available online 28 December 2020

Keywords:
Change impact analysis
Change proneness
Instability
Changeability
Amount of change

ABSTRACT

Change Impact Analysis (CIA) is the process of exploring the tentative effects of a change in other parts of a system. CIA is considered beneficial in practice, since it reduces cost of maintenance and the risk of software development failures. In this paper, we present a systematic mapping study that covers a plethora of CIA methods (by exploring 111 papers), putting special emphasis on how the CIA phenomenon can be quantified: to be efficiently managed. The results of our study suggest that: (a) the practical benefits of CIA cover any type of maintenance request (e.g., feature additions, bug fixing) and can help in reducing relevant cost; (b) CIA quantification relies on four parameters (instability, amount of change, change proneness, and changeability), whose assessment is supported by various metrics and predictors; and (c) in this vast research field, there are still some viewpoints that remain unexplored (e.g., the negative consequences of highly change prone artifacts), whereas others are over-researched (e.g., quantification of instability based on metrics). Based on our results, we provide: (a) useful information for practitioners—i.e., the expected benefits of CIA, and a list of CIA-related metrics, emphasizing on the provision of a detailed interpretation of their relation to CIA; and (b) interesting future research directions—i.e., over- and under-researched sub-fields of CIA.

1. Introduction

Change Impact Analysis (CIA) is the process of investigating the undesired consequences of a change in a software module (Bohner, 2000); and is considered of paramount importance, in the sense that it aims to reduce the *risk of software failure* and the *maintenance cost*.

With respect to the aim of ***reducing the risk of software development failures***, project managers can invest on effective CIA to mitigate the negative consequences of a change (Aljohani and Qureshi, 2016). In traditional risk management, risks are assessed by estimating two parameters (Boehm, 1991): the probability of a risk to occur; and the impact that the occurrence of a risk will have. Tailoring this definition to fit software changes, by considering them as possible risks, the above parameters (namely: ***change impact parameters***) can be interpreted as follows: (*a*) the probability of a software artifact to change; and (*b*) the effort required for undertaking the change. These parameters can be further decomposed as follows—parameter *a* is decomposed to *a1* and *a2*; whereas parameter *b* is decomposed to *b1* and *b2*, respectively:

(a_1) *change proneness*, which is the probability of a software artifact to change—e.g., due to bug fixes, changing requirements (Jaafar et al., 2014);

(a_2) *instability*, which is the probability of a software artifact to change due to changes in other artifacts of the system (ISO/IEC 9126-1:2001)

(b_1) *amount of change* is the extent of changes that are made on a software artifact (Arisholm et al., 2001);

(b_2) *changeability* is the ease of performing changes to a software artifact (ISO/IEC 9126-1:2001).

Additionally, with respect to ***maintenance cost***, CIA can be useful both before and after the application of the change. Before the application of the change, CIA can be useful for effort estimation. For example, knowing how many classes will need to be updated is an indicator of maintenance effort (Haney, 1972)—related to *change proneness*, *amount of change* and *changeability*. After the application of a change, CIA can be useful for test case prioritization. For instance, being aware of co-changing

☆ Editor: [BURAK TURHAN].

* Corresponding author.
E-mail addresses: mkretsou@gmail.com (M. Kretsou), e.arvanitou@uom.edu.gr (E.-M. Arvanitou), a.ampatzoglou@uom.edu.gr (A. Ampatzoglou), ignatios@it.teithe.gr (I. Deligiannis), vgerogian@uth.gr (V.C. Gerogiannis).

https://doi.org/10.1016/j.jss.2020.110892
0164-1212/

requirements is an efficient way to prioritize test cases (Rovegard et al., 2008)—related to *instability*. Thus, effective change impact analysis can be an important factor for reducing maintenance cost, which is often substantial along the software lifecycle—i.e., the total maintenance cost is estimated to comprise at least 50% of total lifecycle costs (van Vliet, 1993).

To visualize the context of CIA, in Fig. 1, we present how the change impact parameters are used upon a change request. The process starts by receiving a maintenance request, which can be classified into four different categories: (a) code enhancements; (b) bug fixes; (c) feature requests; and (d) refactoring suggestions (Palomba et al., 2018). Next, the software developer needs to perform CIA, and get insights on which classes will need to change (*change proneness*), which other classes will need to be co-maintained (*instability*), and how much effort he/she will need for resolving the corresponding request (*changeability* and *amount of change*). Based on this information, the developer is expected to apply the change more efficiently, in terms of time required, number of introduced bugs, etc.

Given the importance of change impact analysis, in this study we aim to provide an overview of the state-of-the-art on this domain, through conducting a Systematic Mapping Study. The mapping study has been designed based on three goals, as described below:

(g1) ***Practitioners' Benefits from CIA:*** The exploration of how CIA facilitates the application of each type of change can be potentially useful for practitioners, in the sense that our systematic mapping study highlights the practical benefits resulted from performing CIA.

(g2) ***CIA Parameters' Quantification***: Additionally, to achieve efficient change management; practitioners should be equipped with established quantification approaches.[1] The change impact parameters can be assessed either *directly* or *indirectly*. As *direct* assessment, we refer to software metrics or methods that could calculate the aforementioned change impact parameters; whereas, as *indirect* assessment, we refer to the use of existing metrics that can be used for proxying their values. Therefore, through this mapping study, we aim at providing a detailed panorama on the existing practices for quantifying change impact parameters.

(g3) ***Research Goals***: Finally, a common goal of secondary studies is the identification of the mostly researched sub-areas of a field, and the identification of possible gaps. Thus, in this study, we aim at providing an overview of the research goals of primary studies, to fulfill the aforementioned expectation.

The main findings of this study are presented as a synthesized list of benefits for practitioners (by applying CIA) for each type of maintenance request, a list of quality properties and metrics that can be used for performing CIA, and a view of under- and over-studied themes in this research area. Based on the aforementioned results, we are able to provide a detailed discussion on the implications of this study for practitioners and a research roadmap for change impact analysis. The rest of the paper is organized as follows: In Section 2, we discuss related work that is relevant to change impact analysis, whereas, in Section 3, we present the adopted systematic mapping protocol. Next, in Section 4 we present the results of our study, and in Section 5 we provide a research roadmap. Finally, in Section 6, we present threats to validity, and in Section 7 we conclude the paper, focusing primarily on the actionable outcomes of this study for practitioners.

[1] One of the most well-known quotes in software engineering, the one by de Marco (1986), suggests that "*you can't control what you can't measure*". In other words, you cannot assess improvement in one aspect, if there is no way to quantify it.

2. Related work

In this section, we present related work: i.e., secondary studies (i.e., Systematic Literature Review—SLR, or Systematic Mapping Study—SMS) that are directly or indirectly comparable to ours. First, we present the two directly related secondary studies, i.e., those that focus on change impact analysis, and next, studies that focus on maintainability prediction, i.e., studies that are related to the change impact parameters (indirect related work). We note that from this section, we have excluded studies that focus on specific technologies or application domain (e.g., Alam et al., 2015a,b; Brink et al., 2016; Saraiva et al., 2012, respectively) in the sense that results and research methods are not comparable. Finally, in the end of this section, we provide an overview of the comparison between our and related work.

Change Impact Analysis. Malhotra and Bansal (2016) performed a literature review on change prediction—i.e., if an artifact is going to change or not. In particular, the goal of this study was to identify: (a) the goal of each study; (b) the types of dataset that are used for prediction; (c) the kind of metrics that are used in the prediction of changes; (d) the usefulness of machine learning methods; and (e) the most popular journal in the area. The search process is conducted between 1998 and 2011. The authors have classified the papers into two categories based on the type of metrics used: papers which have used class level metrics (9 studies) and papers which have not used class level metrics (12 studies). The results suggest that object-oriented measures have a strong predictive power on the phenomenon. In a follow-up study, Malhotra and Khanna (2019) performed a systematic literature review to compare the capabilities of existing software change prediction (SCP) models and evaluate their effectiveness. The study focused on identifying: (a) the predictors that are useful for developing SCP models; and (b) the experimental settings, the categories of data analysis algorithms, the statistical tests and the threats with respect to SCP studies. The search process was conducted between 2000 and 2019 in five DLs (namely Scopus, ACM, Wiley, IEEE, and SpringerLink), identifying 38 primary studies. The results suggested that structural metrics, and in particular CK metrics, have been widely used in SCP models. However, the validation of process metrics and their combination with product metrics is limited in this domain. The main difference of both works compared to ours is that our study is more comprehensive in the sense that it focuses on all change impact parameters and not only the change proneness, i.e., it also explored instability, change amount and changeability.

(Li et al., 2013) conducted a literature review of code-based change impact analysis techniques. More specifically, the study focused on identifying: (a) the techniques for performing CIA; (b) the properties that could characterize code-based CIA; (c) the key application areas; and (d) future work. The search process is performed in four digital libraries (namely ACM, IEEE, ScienceDirect, and SpringerLink) from 1997 to 2010. After applying the selection criteria, 30 studies were identified. The results of their study revealed 23 different CIA techniques. Additionally, the authors provide a framework that the practitioners could identify and compare different CIA techniques, based on the specific needs of the practitioner, whereas researchers could use the suggested framework to develop new techniques. Alam et al. (2015a,b) performed a systematic literature review to explore the impact analysis and change propagation in Business Process

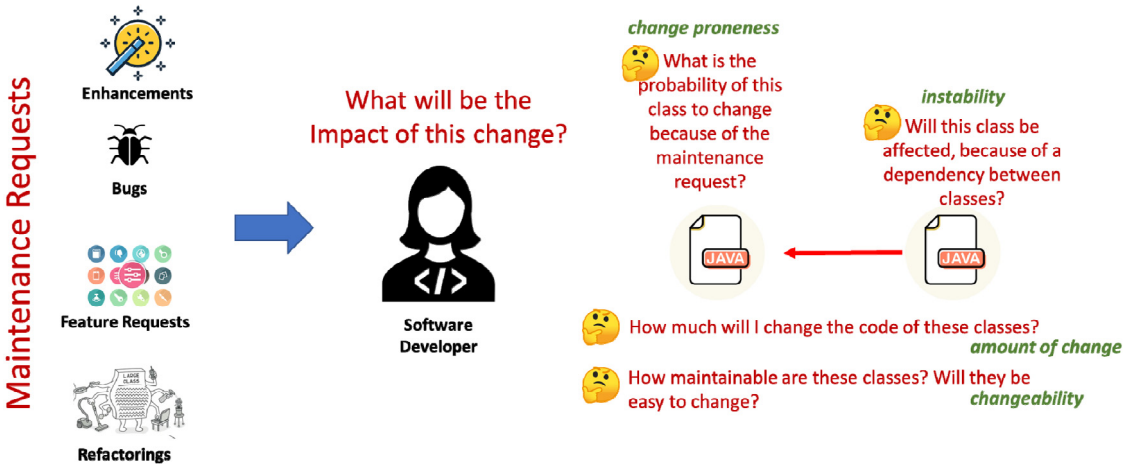

Fig. 1. Change impact analysis process.

Management (BPM) and Service-Oriented Architectures (SOA). In particular, the study focused on: (a) identifying changes and dependencies across different abstractions layers; and (b) classifying propagation techniques and change analysis in two domains. The search process was applied on six DLs (namely: ACM, IEEE, ScienceDirect, SpringerLink, Wiley, and Emerald) from 2007 to 2014. At the end of the selection process 60 primary studies were retained for further analysis. The results of the study suggested that dependency analysis is the most frequently adopted technique followed by traceability. Additionally, further categorization of dependency analysis indicates that graph-based techniques are extensively used, followed by formal dependency modeling. The majority of change propagation solutions are top-down and semi-automated. Moreover, there are no mature tools and techniques to provide end-to-end change analysis and propagation support. The difference compared to our work is that Li et al. (2013) and Alam et al. (2015a,b) do not deal with quality metrics or change impact parameters. Additionally, Li et al. (2013) focus only in techniques that are based on source-code and Alam et al. (2015a,b) focus only in two specific domains, whereas in our work we focus on studies, which can be applied to any type of software artifact and generic domain. On top of that Alam et al. (2015a,b) considered any kind of dependency analysis as change impact analysis, which if generalized outside BPM and SOA, would lead to a tremendous amount of studies that focus on the coupling between classes.

Maintainability Metrics. Arvanitou et al. (2017a) conducted a mapping study to investigate design-time quality attributes and metrics. In particular, the authors explored: (a) the most important quality attributes for each application domain and development phase, and (b) the use of quality metrics for assessing each quality attribute. For the quality metrics, the authors identified if the quality metrics use a formula for quantifying quality metrics, the empirical evidence of each metric and if there is tool support for automatically calculating them. The search strategy identified papers until 2016 and was conducted on 12 specific venues. At the end of the selection process, 154 primary studies were selected. The results of the study suggest that maintainability is the most commonly studied quality attribute, regardless of the application domain or the development phase. Additionally, quality properties (e.g., cohesion, coupling, etc.) are more frequently studied than quality attributes (e.g., maintainability, reusability, etc.) and quality attributes is performed by a single metric rather than a function of multiple metrics, and quality metrics are mostly validated in an empirical setting.

Riaz et al. (2009) presented a systematic review of software maintainability prediction and metrics. More specifically, the study focused on identifying forecasting methods/techniques for maintainability prediction and the level of evidence in these methods. The search process was conducted in 9 digital libraries (namely Scopus, IEEE, Current Contents and Computer Database, ScienceDirect, SpringerLink, Inspec, ACM, and ProQuest Computing) from 1985 to November 2008. After applying the selection criteria, 14 studies were selected. The results of this study suggest that the most important predictors were those based on size, complexity and coupling at source code level. Another outcome of this study is that the level of evidence was found limited for validating maintainability prediction techniques compared to the models of van Koten and Gray (2006), and Zhou and Xu (2008). Additionally, Jabangwe et al. (2014) performed a literature review to (a) identify object-oriented measures at the source code level, which they have been empirically evaluated, and linked to external quality attributes, and (b) evaluate the consistency of the link between them across studies. The search strategy was conducted in five digital libraries (ACM, IEEE, Scopus, Compendendex and Inspec) until 2012. After the selection criteria, the authors were selected 99 studies. Then, Jabangwe et al. (2014) focused on four specific quality attributes: reliability, maintainability, efficiency and functionality. The results suggest that the most commonly studied quality attribute is maintainability, which in most of the cases is quantified through the Chidamber and Kemerer (CK) metric suite (1994). The studies of Riaz et al. (2009), and Jabangwe et al. (2014) are related to ours; however, they both focus on maintainability rather than change impact analysis, i.e., they are broader in scope. Finally, Saraiva et al. (2012) performed a mapping study to investigate which metrics can be used to measure the maintainability of software developed with Aspect-Oriented Programming (AOP). The search strategy identified papers until June 2011 and was conducted on four DLs (IEEE, ACM, Compendex and ScienceDirect). At the end of the study identification process, 138 primary studies were selected. The results proposed a catalog that can guide researchers in selecting metrics that are suitable for their studies. The differences of this work compared to our study is that Saraiva et al. (2012) focus on a specific programming paradigm (i.e., AOP) and a specific quality attribute (i.e., maintainability).

Malhotra and Chug (2016) conducted a systematic literature review in the field of software maintainability to identify important aspects which could affect maintenance effort. More specifically, the study focused on identifying techniques, metrics, and tools that are related with maintainability. The search strategy was conducted in nine DLs (namely Google Scholar, Scopus, ScienceDirect, Springer, ACM, IEEE, Wiley, Web of Science and Compendex) from 1991 to 2015: retaining 96 primary studies. The

results suggested that design metrics are still the most favored option for capturing the characteristics of any given software, and in particular the metrics suites proposed by Chidamber and Kemerer (1994) and Li and Henry (1993). Finally, Benestad et al. (2009) performed a literature review on change-based studies (i.e., analyze data that describe the individual changes that are made to software). More specifically, the goal of this study was to identify change attributes and change measures that drive and predict costs and risks during maintenance and evolution. The search strategy was applied on two DLs (namely: Google Scholar and IEEE) from 1993 to 2007. As an outcome, 34 primary studies were selected. Benestad et al. (2009) proposed a conceptual model for change-based studies that enables them to classify the attributes. The main differences of our study compared to Benestad et al. (2009) is that our study does not aim at providing a classification of changes and their characteristics, but on quantifying the important parameters while applying these changes.

Comparison to Related Work. Next, we present the comparison between the related studies and our work, in terms of: *amount of studies*, *covered period*, and *research contributions*. For each study we list the research method that they have used, the direct or indirect relation to CIA (as indirect we consider broader studies, e.g., on maintainability, that cover some change properties), the number of included papers, the period covered, and the overlap with our goals (see Table 1).

On the one hand, given Table 1, we can observe that our study is a valuable extension of the research-state-of-the-art, by considering the ***amount of studies*** and the ***covered period***. In particular, among directly comparable studies: two have finished the data collection around 2010 (from 2010 and on, 80% of the primary studies of our dataset were published after 2010), one study around 2015 (from 2015 and on, 45% of the primary studies of our dataset were published after 2010), and one study with data collection ending on 2019 (consisting this study up-to-date). Nevertheless, the scope of the review of Malhotra and Khanna (2019) was substantially narrower in scope, since it aimed only at change proneness. Thus, the amount of studies that we considered is almost doubled up. On the other hand, regarding ***research contributions***, based on Table 1 and the previously presented annotated bibliography, our study is the only up-to-date secondary that discusses the practical benefits of change impact analysis, constituting the study as highly relevant for practitioners. We note that the study of Li et al. (2013) had a similar goal, but captured only a small fraction of primary studies, published until 2010. Additionally, our study is the only one that discusses simultaneously all change impact parameters (i.e., change proneness, instability, amount of change, and changeability), enabling a fair comparison in terms of the research load for each parameter, and the identification of future research directions.

3. Study design

This section presents the protocol of the systematic mapping study. A protocol constitutes a pre-determined plan that specifies the research questions and how the mapping study has been conducted. Our protocol is presented according to the guidelines suggested by Petersen et al. (2008).

3.1. Objectives and research questions

The primary goal of this study, stated using the Goal-Question-Metrics (GQM) format (Basili et al., 1994) is to: ***analyze*** existing change impact analysis methods for the ***purpose*** of characterization ***with respect to*** the change impact parameters (namely: change proneness, instability, amount of change, and changeability) ***from the point of view of*** researchers and practitioners ***in the context of*** software maintenance. Based on the aforementioned GQM formulation and the goals stated in Section 1, we have set the following research questions:

RQ_1: *What are the benefits for performing Change Impact Analysis for practitioners?*
RQ_1 is related to the usefulness of CIA methods to practitioners. In particular, we explore the motivation of primary studies and identify the reasons for which practitioners deem CIA as important. Answering this research question will shed light on the types of changes that can be supported by CIA methods.

RQ_2: *How can change impact parameters be assessed?*
RQ_2 aims at exploring change impact parameters quantification. In particular, we aim at highlighting: (a) the most studied change impact parameters for each development phase, (b) the most used software artifacts for the quantification of each CIA parameter; (c) the most important metrics for quantifying each CIA parameter directly; and (d) the most important metrics for indirectly quantifying each CIA parameter.

RQ_3: *What is the goal of researchers when setting up their primary studies?*
RQ_3 is related to the research goals of studies related to CIA. In particular, we focus on researchers and investigate the goals of the primary studies so as to extract: (a) the most studied research sub-areas; and (b) possible research gaps that deserve future investigation.

3.2. Search process

Publication Venue Selection. We defined our search strategy by considering the goal and research questions of the study. In particular, we opted for performing an automated search, through digital libraries (DL) portals, on specific publication venues. The reasoning behind this decision is our intention to retrieve only primary studies that are of guaranteed top-quality—such a choice is well acknowledged as a best practice in software engineering secondary research (Kitchenham et al., 2009a,b). Despite the fact that the assessment/controlling of the quality of the primary studies is not a prerequisite for mapping studies, we have preferred to focus on top quality venues for two reasons:

- ***Studying a Broad Research Area***. According to the recent guidelines on how to identify and report threats to validity for secondary studies (Ampatzoglou et al., 2019): "*In case the research team is investigating a very broad topic, or is interested in including only top-quality venues, venue selection processes are described in Cai and Card (2008), Galster et al. (2014) and Kitchenham et al. (2009a).*". Therefore, our decision is reasonable, since our study covers a very broad topic, which would be unmanageable if we targeted complete databases.
- ***Mitigation of Data Validity Threat***: Additionally, even for SMS, the quality of the primary studies is an important factor for the quality of the secondary study. As explained in the guidelines on managing threats to validity for secondary studies (Ampatzoglou et al., 2019), the selection of top-quality venues is the top mitigation action for the threat to validity: "*Quality of Primary Studies*" categorized under "*Data Validity*".

The venues have been selected based on the study of Karanatsiou et al. (2019), which is the latest article of the well-known series

Table 1
Related work overview.

Reference	Research method	Relation to CIA	#papers	Period	Practice (goal-1)	Parameters (goal-2)	Research (goal-3)
Malhotra and Bansal (2016)	SLR	Direct	21	1998–2011		Change proneness	X
Malhotra and Khanna (2019)	SLR	Direct	38	2000–2019		Change proneness	X
Li et al. (2012)	SLR	Direct	30	1997–2010	X		X
Alam et al. (2015a,b)	SLR	Direct	60	2007–2014			X
Arvanitou et al. (2017a)	SMS	Indirect	154	Until 2016		Change proneness instability changeability	
Saraiva et al. (2012)	SMS	Indirect	138	Until 2011		Change amount	
Riaz et al. (2009)	SLR	Indirect	14	1985–2008		Amount of change	
Jabangwe et al. (2014)	SLR	Indirect	99	Until 2012		Change proneness changeability amount of change	
Malhotra and Chug (2016)	SLR	Indirect	96	1991–2015			
Benestad et al. (2009)	SLR	Indirect	34			Changeability amount of change	
Our study	SMS	Direct	111	Until 2019	X	Change proneness instability changeability change amount	X

Table 2
Selected venues.

Publication venue	DL
Transactions on Software Engineering (TSE) International Conference on Software Engineering (ICSE) Symposium on Empirical Software Engineering and Measurement (ESEM) International Conference on Automated Software Engineering (ASE) International Conference on Software Processes (ICSP) International Conference on Software Analysis, Evolution and Reengineering (SANER)/previously CSMR and WCRE International Conference on Software Maintenance and Evolution (ICSME) IEEE Software (SW)	IEEE
Transactions on Software Engineering and Methodology (TOSEM) International Symposium on the Foundations of Software Engineering (FSE)	ACM
Empirical Software Engineering (ESE)	Springer
Software: Practice and Experience (SPE) Journal of Software: Evolution and Process (JSEP)/previously JSME	Wiley
Information and Software Technology (IST) Journal of Systems and Software (JSS)	ScienceDirect

of bibliometric papers for top-scholars and institutes in software engineering. The venue selection process is based on four criteria; we selected venues that: (a) are classified as "Computer Software" by the Australian Research Council with an evaluation higher than or equal to level "B" for both journals and conferences; (b) are strictly relevant to the software engineering domain; (c) on average have more than 1 citation per month, per published article; and (d) are general-scope journals, not restricted to phases or activities—with the only exception to maintenance venues, which are of special interest to this study (e.g., CSMR/WCRE, ICSME, SANER, JSME). In Table 2 we present the selected venues.

Search String Construction—Study Identification. Next, we applied our search string (see box below) on the full-text of candidate primary studies, published in the aforementioned venues. The goal of this step was to return studies that are relevant to change impact analysis. To construct the search string, we have followed a systematic process. First, based on the goals of this study, we have split our search string into two components: the first one is related to change impact analysis and its parameters (see Section 1), whereas the second has been added due to RQ_2, which restricts our goal to studies that propose, use, or evaluate a metric or a method for CIA. An alternative on the first part, would be the use of the term "*software change*", but applying this term on the full text would return many irrelevant results, not being able to contribute to answering the set research questions. We note that we have piloted this decision on the first 5 pages on Google Scholar, verifying our belief that broadening the search string would needlessly increase the amount of retrieved studies (in the sense that the majority of these studies would be excluded in later stages). Regarding the second part, we used two alternatives for measurement and the term "*method*" to retrieve studies that assess CIA, but not through metrics. By considering that the search was conducted in the full text of the publication, we strongly believe that these terms would appear at least ones in a relevant manuscript. To validate the final version of the search string, we have performed a piloting before applying it to all venues. In particular, we have checked that all primary studies of a broader secondary study (i.e., (Arvanitou et al., 2017b)) published in three venues (namely TSE, IST, and JSS) have been retrieved from applying our search string.

((“change impact analysis” OR “change proneness” OR “changeability” OR “instability” OR “change amount”) AND (“metric” OR “method” OR “measurement”))

Although no publication indexing sites (e.g., Scopus, Google Scholar, etc.) have been used, we have performed this step, since some conferences publish their papers in more than one digital library, e.g., ESEM conference is hosted in ACM and IEEE DLs.

Studies Filtering Phase. The next step of the process was to identify all the primary studies that are relevant to this mapping study. To this end, we have set several inclusion and exclusion criteria, applying a systematic process. The definition of the inclusion criteria has been based on the goal of the study: first, it was mandatory to assess the paper as relevant to CIA. Second, there was a need for the discussion around the metric/measure process to be central in each candidate primary study. The definition of exclusion criteria followed the most classic ones from the literature. Studies to be included in the final dataset had to satisfy the first Inclusion Criterion (IC) and one or more of the rest ICs, whereas at the same time, they were not satisfying any Exclusion Criteria (EC):

IC1 AND (IC2 OR IC3) AND NOT (EC1 OR EC2)

The inclusion criteria of our systematic mapping study are:

- IC1: The study is related to change impact analysis;
- IC2: The study defines one or more change impact parameters;
- IC3: The study defines one or more quality metrics;

The exclusion criteria in our mapping study are:

- EC1: The study is written in a language other than English;

- EC2: The study is an editorial, invited/position/opinion paper, keynote, tutorial, poster or panel.

The article filtering phase has been handled by the first three authors of this study, using the voting method, as described by Farhoodi et al. (2013). The first three authors inspected the publication's full text and assigned a vote on a 4-point scale (4: strong inclusion, 1: string exclusion)—leading to a maximum score of 12 points. Following the threshold used by Farhoodi et al. (2013), we retained studies with a score higher to 8 points. Studies that were marked with exactly 8 points (12 in total), were reviewed and discussed with the four and fifth authors of the study. To ensure that the researchers involved in the data collection shared a common understanding of the inclusion criteria, first a thorough discussion among authors was performed. Next, we piloted the first 30 papers, which have been assessed in pairs by the four authors so as to have an open discussion on the voting scores. All authors explained their scores, until a consensus was reached. The high degree of a common understanding on the criteria is supported by the low disagreement rate in the inclusion exclusion phase (i.e., 1.7%). We note that the exclusion criteria (language and type of paper) are straightforward and no validation or piloting was required.

Search Process Overview. In Fig. 2, we present an overview of the search and filtering process along with the number of studies at each phase. At the end, we have retained 99 primary studies to be included in this mapping study, and proceed with data collection—see Appendix A.

3.3. Data collection

As part of data extraction, for all included studies, we have defined a set of variables that describe each primary study. Thus, for every study, we have recorded the values of the following variables:
[V1] **Publication Title**
[V2] **Author**: List of authors
[V3] **Year**: Publication year
[V4] **Type of Paper**: Conference or journal
[V5] **Publication Venue**: Name of the conference or journal
[V6] **Benefit** from performing CIA (e.g., reduce debugging time, reduce time to add feature, etc.)
[V7] **Research Goal** set in the primary study (e.g., propose or validate a novel CIA metric, etc.)
[V8] **Development Phase**: Investigated development phase (e.g., requirements, architecture, design)
[V9] **Type of Software Artifact**: Explored software artifacts (e.g., class diagram, use case, etc.)
[V10] **Change Impact Parameters**: Change Proneness, Instability, Changeability, Amount of Change
[V11] **Quality Metrics or Method**: Novel metrics or methods for directly quantifying CIA parameters
[V12] **Predictors of CIA parameters**: Existing metrics for indirectly assessing the CIA parameters

To strengthen the validity of data extraction, we used the following systematic process. The first two authors independently extracted data. If there were inconsistencies in the extracted information, the involved authors discussed the inconsistencies between them. If they were not able to resolve the discrepancies, the third author joined the discussion to resolve the disagreement. During the process 15 inconsistencies have been resolved. The dataset is available online.[2]

[2] https://users.uom.gr/~a.ampatzoglou/aux_material/JSS2020_dataset.xls.

Table 3
Data analysis overview.

Research question	Used variables	Analysis method
RQ_1	[V6]	Open Card Sorting Frequency tables
RQ_2	[V8], [V9], [V10], [V11], and [V12]	Frequency tables Cross tabulation
RQ_3	[V7] and [V10]	Open Card Sorting Frequency tables Cross tabulation

3.4. Data analysis

Variables [V1] – [V5] have been used for documentation purposes. The rest of the variables have been used for answering the research questions and describing the context of the study. For reporting purposes, we used common visualization methods (e.g., bar charts, pie charts, etc.), frequency tables, and cross-tabulation of variables. Also, for consolidating the values of variables retrieved from different studies, we have employed the Open Card Sorting methodology (Spencer, 2009). An overview of the data analysis overview, presenting the mapping between collected variables, research questions, and data analysis methods, is provided in Table 3.

During data analysis, as far as variables [V6] and [V7] are concerned, we have noticed that the terminology used in the various identified primary studies was quite diverse. In particular, we have applied the Open Card Sorting methodology (Spencer, 2009): (a) recorded themes from the research goals as identified in the primary studies (without any processing); (b) reviewed the themes to find candidates for merging; and (c) defined the names of the final themes. The first and the second author performed the process of identifying the themes, and the third, four, and fifth authors validated the results. During the consolidation process on the themes' extraction and their naming (i.e., [V6]), there were some disagreements (approximately 6%), which have been resolved by a discussion among the authors. On the other hand, regarding since the naming of the themes and the distinction was very clear from early in the data extraction process, the amount of conflicts was very limited (<2%). Finally, since [V12] was expected to lead to a vast number of existing metrics, as part of meta-analysis, we recorded the quality property (e.g., coupling, cohesion, complexity, inheritance, etc.) that the metric assesses.

4. Results

In this section we present the results of our study organized by RQ. In Section 4.1, we discuss our findings related to the benefits of CIA for the practitioners. In Section 4.2, we provide the most studied change impact parameters and the proposed methods for assessing them (directly or indirectly). Finally, in Section 4.3, we present researchers-related results, i.e. most studied (and understudied) sub-areas.

Initially, we provide some descriptive statistics (using frequencies) for the dataset of primary studies. Based on the selection process, we have retrieved 111 primary studies. Fig. 3 illustrates the number of studies published per 5-year periods: we can observe that after 2009 the number of studies has increased substantially. Thus, in the last decade, researchers try to explore methods for performing CIA. Additionally, in Table 4 we present the frequency of study per publication venue. We observe that from the 111 primary studies, 83 studies are published in journals, whereas 27 in conferences. Out of the 111 publications, 20 have been published in maintainability-related venues (i.e., SANER, ICSME, and JSEP), suggesting that the topic is not

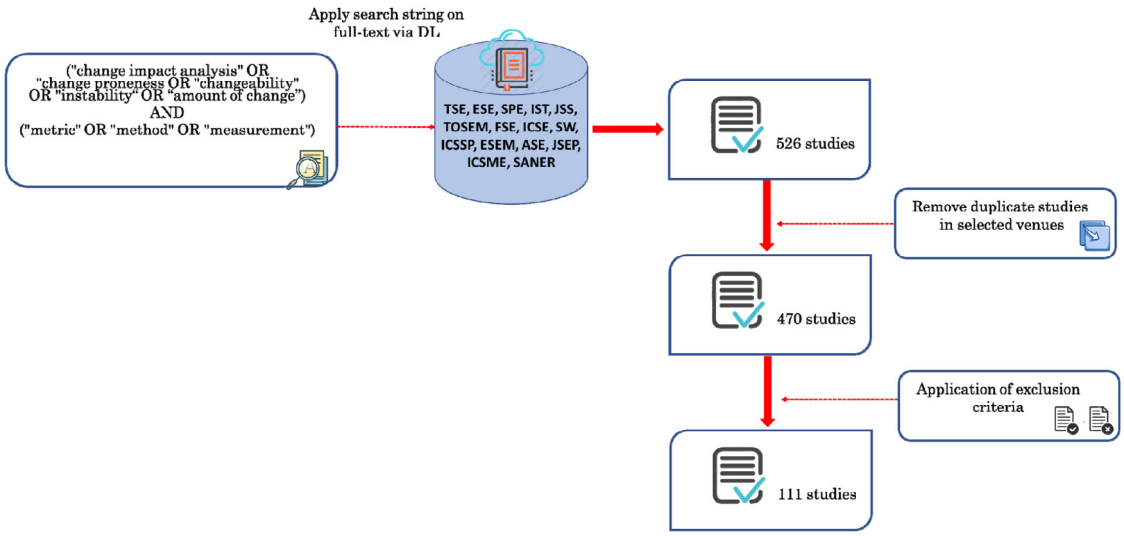

Fig. 2. Overview of search and filtering process.

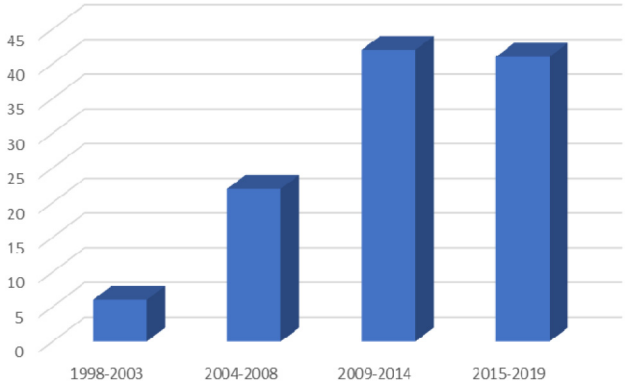

Fig. 3. Publications years' frequency.

Table 4
Publication venues.

Publication venue	#Studies
ESE	34
IST	19
JSS	14
JSEP/JSME	7
SANER/CSMR/WCRE	7
TSE	6
ICSME	6
ICSE	5
ESEM	6
SPE	3
ASE	2
FSE	1
SW	1

restricted to a dedicated community, but is deemed as important to the whole software engineering community. We note that since the number of studies is 111, there is no need to represent data as percentages: the absolute number and the percentage are very close.

4.1. Practitioners' benefits from performing change impact analysis (RQ_1)

In this section, we present the results of our mapping study related to the usefulness of performing CIA from the point of view of practitioners. Out of this process, we have identified that 89 studies (80%) report an industrial motivation or implications for practice, implying a high industrial relevance for these studies (Ivarsson and Gorschek, 2011). Table 5 lists the most reported benefits of CIA in the primary studies, organized into five (5) themes, based on the outcome of the Open Card Sorting synthesis process, described in Section 3.4 (Spencer, 2009). The vast majority of the studies were classified by using four themes having identical names with the four software maintenance types proposed by van Vliet (1993): Adaptive,[3] Corrective,[4] Preventive,[5] and Perfective[6] Maintenance, while there were only 4 primary studies which have been classified to a fifth theme that was relevant to Reuse.

Based on Table 5, we can observe that the dominant theme of the reported benefits is *Adaptive Maintenance* (47%), followed by *Perfective Maintenance* (17%) and *Corrective Maintenance* (9%). *Preventive Maintenance* benefits are discussed in 6% of the studies, whereas *Reuse* potential in 3%. The interpretation of how CIA leads to these benefits and the rationale for the classification are justified below:

- ***Adaptive Maintenance***. The main benefit in this category is the "*Improvement of software maintenance tasks*". This benefit is obtained in terms of maintenance efficiency, since every time that a new request arrives in the software development company, the developer is aware of co-changing artifacts; i.e., there is no need to identify which parts of the system need update. In the relevant literature, the decrease of this mental process is reported to be more time-consuming, compared to the application of the change per se (Kosti et al., 2018). Additionally, CIA can aid developers in the "*Improvement of the accuracy of effort estimation*": by applying CIA, developers are not aware only of local changes in the artifact to be updated, but also on the possible ripple effects, which may increase maintenance costs up to 75% (Galorath, 2008; Chen and Huang, 2009). Finally, the "*Identification of cross-cutting concerns*", which can be achieved by identifying frequently co-changing artifacts (or system-wide changes) is also useful, in the sense that architectural

[3] Tasks related to the addition of new features, the migration to a new runtime environment, etc.

[4] Tasks related to the fixing of bugs identified by the end-users/customers.

[5] Tasks related to the identification of bugs, before the end-user.

[6] Tasks related to the improvement of the system quality.

Table 5
Frequency of the benefits obtained by performing CIA.

Benefits	Themes	#Studies
Improvement of software maintenance tasks	Adaptive maintenance	50
Reduction of effort to refactor or Improve quality	Perfective maintenance	13
Identification of fault-prone artifacts	Preventive maintenance	7
Reduction of debugging time	Corrective maintenance	6
Identification artifacts that are difficult to maintain	Perfective maintenance	6
Improvement of reuse opportunities	Reuse	4
Provision of assistance along test-case selection	Corrective maintenance	3
Reduction in the number of introduced bugs	Corrective maintenance	2
Improvement of the accuracy of effort estimation	Adaptive maintenance	1
Identification crosscutting concerns	Adaptive maintenance	1

changes (i.e., large-scale changes that are applied system-wide) are usually costlier compared to local ones (Brown et al., 2011); therefore, being aware of such changes can lead to their efficient management.

- ***Perfective Maintenance***. The benefit obtained by CIA in terms of perfective maintenance, is two-fold: related to the identification of design hotspots (i.e., parts of the system that urge for quality improvement) and to the ease of applying the quality optimization per se. On the one hand, the "*Reduction of Effort to Refactor or Improve Quality*" is the most studied benefit, which among others, is achieved by the fact that developers are aware of the test that need to be executed upon refactoring—that could be violated due to ripple effects (Kabaili et al., 2005). On the other hand, the "*Identification Artifacts that are Difficult to Maintain*" is assisted by CIA, and in particular by the effort-related parameters: *changeability* and *amount of change*. Being aware of the artifacts that are difficult to maintain, can lead to a refactoring prioritization aiming at improving aspects of quality (e.g., coupling, cohesion, complexity, etc.) that are related to maintainability—a notion that is heavily exploited in the technical debt community, through the concept of interest probability (Arvanitou et al., 2017b).
- ***Corrective Maintenance***. In terms of corrective maintenance, the instability CIA parameter can aid in the "*Reduction of Debugging Time*", since by performing CIA, the developer can be aware of the classes that need to change along a set of bug fixing activities (due to possible ripple effects). At minimum (even if a change does not need to be applied to other artifacts), the developer gets informed on a limited number of tests that need to be executed through the "*Provision of assistance along test-case selection*". Being aware of the tests that need to be executed (e.g., in a regression testing phase) is discussed by Kabaili et al. (2005), along with the relation of this task with the existence of ripple effects/instability. A consequence of the aforementioned benefits is the "*Reduction in the Number of Introduced Bugs*", due to the accurate execution of all required tests, guarantees (to some extent) the correct application of the software.
- ***Preventive Maintenance***. With respect to preventive maintenance, we have been able to identify one related benefit, namely "*Identification of Fault-Prone Artifacts*"; which however, is referred in seven (7) identified studies. CIA can help in identifying fault-prone artifacts by exploiting the change proneness of artifacts, i.e., more change prone classes are usually more error-prone as well (Khomh et al., 2012). Being aware of the fault-prone artifacts may indicate the necessity of introducing preventive/internal testing procedures to those that have the higher probability to produce errors.
- ***Reuse***. Finally, CIA can aid software engineers through the "*Improvement of Reuse Opportunities*". More specifically, ad/hoc reuse practices, aim at the identification of reusable sets of artifacts, based on dependency analysis (Ampatzoglou et al., 2012). Being aware of which artifacts need to be reused along with the targeted artifact (to minimize the adaptation time), can be guided by instability and ripple effect analysis, in the sense that they are both relying on artifact dependencies (Arvanitou et al., 2015).

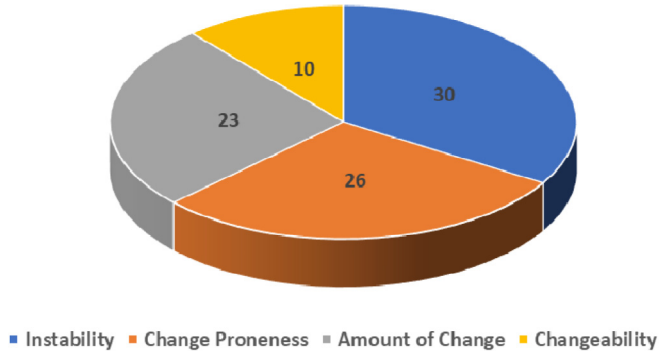

Fig. 4. Frequency of change impact parameters.

4.2. Quantification of change impact parameters (RQ_2)

In this section, we focus on the quantification of change impact parameters; thus, we focus only to studies that involve either metrics/methods [V11] or predictors [V12] of a CIA parameter. Initially as a demographic analysis, in Fig. 4, we present the number of studies in which each change impact parameter has been assessed.[7] Based on Fig. 4, the most studied change impact parameter is *instability*, followed by *amount of change* and *change proneness*. An interesting observation from Fig. 4, is that the change impact parameters that are related to the risk probability (i.e., the probability of an artifact to change) are over-studied (∼63%), whereas only 37% focuses on the impact of the risk (i.e., how large chunks of code are going to be changed). This finding can be attributed to the fact that the uncertainty of an event to occur is higher compared to the uncertainty of the extent of the phenomenon. Next, we discuss: (a) the ways of quantifying the change impact parameters: and (b) the study of CIA parameters quantification in various development phases/artifacts.

Change Impact Parameters Quantification. As mentioned in Section 1, change impact parameters can be quantified directly or indirectly. On the one hand, regarding the *direct quantification* of change impact parameters, the analysis process has led to various and numerous metrics, whose detailed presentation was not possible in the manuscript. Nevertheless, since we acknowledge that this is a vital information for this study, in Appendix B, we present a glossary including: (a) the proposed metric; (b) its acronym; (c) its calculation method; and (d) a link to the original study.

[7] We note that some studies might refer to more than one CIA parameter.

The glossary is organized into subsections, based on the change impact parameter that they quantify. Below, we discuss the main ways, based on which each CIA parameter can be quantified. We note that while discussing each CIA parameter, we present only one representative reference, since the full list can be found in Appendix B.

- *Instability* is assessed by 21 distinct metrics that can be organized into three categories. The first and dominant category, captures instability as the *percentage of the system that is affected* by a given change—e.g., SDI (System Design Instability) (Alshayeb and Li, 2005). The second category is not calculated at the system level, but on the artifact pair level, denoting *the probability of one artifact to change due to ripple effects*—e.g., IF (Impact Factor) (Sun et al., 2014). The final category is again a measure at system level, denoting the scattering of changes inside files, classes, etc.—e.g., CD (Change Dispersion) (Misirli et al., 2016).
- *Change Proneness* is assessed in the literature by 20 distinct metrics, which can be organized into four main categories. The first category, captures change proneness as a *frequency of commits*/revisions in which an artifact (file, class, method, etc.) has changed—e.g., NPC (Number of Prior Changes) (Misirli et al., 2016). The second category relies on time as unit and calculates the *period of time* in which an artifact remains unchanged—e.g. AA (Average Age) (Moser et al., 2004). The third category unifies metrics from the first two categories. In particular, it normalizes the frequency of commit changes as a *percentage over the commit history*, trying to provide a fair assessment between codebases with historical data at a different level of magnitude—e.g. LIKELIHOOD (Mondal et al., 2018). The fourth category, aims at exploiting the metric of the first category, so as to project them to future commits, acting as predictors of artifact evolution—e.g. Frequencies of Future Changes (Khomh et al., 2019).
- *Amount of Change* is the CIA parameter with the most metrics in the literature (i.e., 25 distinct metrics). Despite their variety most of these metrics are quite similar, and to some extent simplistic (in the sense that they are calculated as simple counts). The metrics are classified according to two factors: (a) the type of change—i.e., add, delete, modify, or both add and delete (churn); and (b) the level of granularity (i.e., components, files, classes, methods, lines of code, etc.). Therefore, combining the values of the above factors, one can identify metrics such as: *Number of Added/Deleted/Modified Modules/Operations/Members/Classes/Files* (Tizzei et al., 2011; D'Amorim and Borba, 2012; Misirli et al., 2016; Stevanetic and Zdun, 2018), *Number of Added/Deleted Lines of Code* (Arisholm, 2006; Hindle, 2015), or *Number of Modified Lines/Total Churn* (Woo et al., 2009). A different line of thinking for quantifying amount of change is the assessment of actual system size (e.g., Class/File/Line Growth (Alshayeb and Li, 2005; Hindle, 2015; Misirli et al., 2016), supposing that a change in these values denotes the extent of changes among different versions of the software.
- *Changeability* has been associated with two distinct metrics: namely, *Effort of Change in Minutes* (Arisholm et al., 2001; Balogh et al., 2015) and *Changeability Index* (Decan et al., 2019). We note that the low number of metrics or methods for directly capturing changeability does not imply that this parameter is not important, but that its direct quantification is very straightforward in an after-the-fact (application of maintenance) analysis, i.e., to record how much time or effort a specific change has taken. However, the prediction of changeability is a very interesting and challenging topic, which has been investigated thoroughly in other secondary studies (Jabangwe et al., 2014; Riaz et al., 2009).

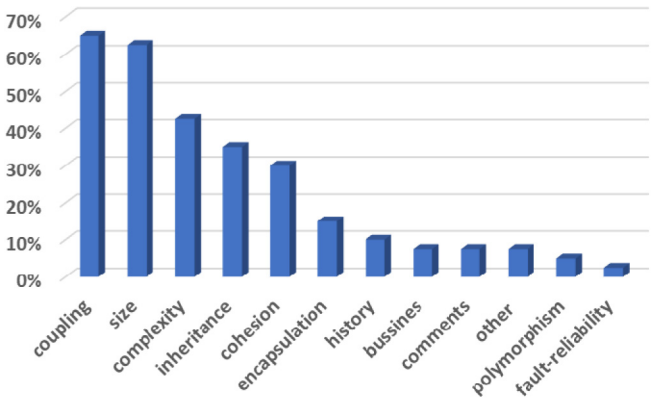

Fig. 5. Frequency of the quality properties.

On the other hand, regarding *indirect quantification*, through existing metrics, we have followed a two-step analysis. First, similarly to before, in Appendix C, we present the full mapping among existing metrics, CIA parameters, and all identified primary studies. Second, we performed a meta-analysis to explore the relation of CIA parameters and the quality properties that the metrics are related to. In particular, Fig. 5 presents the frequencies of the quality properties that are linked (indirectly) to change impact parameters. For instance, at least one coupling metric is used in 26 studies out of the 40 that use indirect CIA parameters assessment (65%). The results suggest that the most studied quality property for quantifying change impact parameters is *coupling,* followed by *size*, *complexity, inheritance,* and *cohesion*. This outcome is somehow expected in the sense that these properties are the most closely related to software maintainability. This finding is also supported by literature: according to Riaz et al. (2009), the most notable maintainability models are using metrics from these quality properties to quantify software maintainability (van Koten and Gray, 2006; Zhou and Leung, 2007). Apart from these dominant properties, other secondary ones have been studied: encapsulation (e.g., number of private/protected methods/attributes); history (e.g., number of commits); business (e.g., number of stakeholders involved in maintenance); comments (e.g., comment density); polymorphism (e.g., number of polymorphic methods); and fault-tolerance/reliability (e.g., number of bugs). We note that "Other" includes results obtained from a limited number of papers, e.g. Power and Malloy (2004) studying the instability and the complexity of grammar-based software applications, such as compilers, editors, program comprehension tools etc.

To proceed to a more fine-grained analysis, in Table 6, we present the results of cross-tabulating CIA parameters and quality properties. In particular, in Table 6, for each pair of CIA parameter and quality property, we provide two values: (a) the number of distinct metrics in the cross-tabulation; and (b) the percentage of studies in which at least one metric of this category has appeared—sorted by (b). For example, given the first row, we can observe that 30 coupling metrics have been reported as relevant to instability, and these metrics span in 71% of studies that use metrics for assessing instability. We note that both views are useful: The first view denotes the availability of metrics (a large number can be perceived both positively and negatively, in the sense that the selection might end up to be confusing), whereas the second view denotes the importance of the property in assessing the specific change impact parameter. The results of Table 5, can be discussed as follows:

- The relation between *coupling* and ***instability*** (coupling is ranked as first both in terms of frequency and absolute

number of metrics), is expected, in the sense that software dependencies are the means for transferring changes from one artifact to another (e.g., through aggregation between two classes) (Arvanitou et al., 2015). The following properties are *complexity* and *size*. The relation between complexity and instability can be ascertained by the fact that a highly complex system might lead to bugs, which are the "cause" of ripple effects (Yau et al., 1978). Furthermore, the relation to size can also be attributed to the indirect relation of coupling (i.e., the more classes exist in the system the highest the average coupling Harrison et al., 1998) and reasons for change (i.e., the more lines of code a class has the more reasons to change Lippert and Roock, 2006). Regarding specific metrics (from Appendix C), we can observe that the most used metrics for assessing instability are two complexity (*WMC* and *CC*) and two size metrics (*LOC* and *NOC*). This is an interesting observation, since coupling metrics are ranked lower than the aforementioned ones. This finding can be explained by the vast amount of distinct coupling metrics, which probably leads to lower frequencies: 7 complexity metrics are studied in 43% of the studies, whereas 29 different coupling metrics are examined in 71% of the studies.

- Regarding ***amount of change,*** the dominant properties are *size*, *inheritance, cohesion* and *complexity*. Since the causal link between the size of an artifact and the amount of change is straightforward (i.e., the larger the size of the artifact the more room there is for artifact modifications), we focus more on the rest three quality properties. First, the extensive use of inheritance often results in larger changes, due to the overlap, overriding, and reuse of source code structures (such as methods, attributes, etc.) (Shaheen and Bousquet, 2009). Second, the relation to cohesion can be explained by the fact that artifacts that are related to more than one functionality (i.e., having lower cohesion), are more prone to change collectively (or at least in larger parts) by a single maintenance ticket, that relied upon multiple axes of change (Martin, 2003). Finally, more complex artifacts (e.g., methods with many control statements) are expected to change in more parts of their implementation. For instance, a method that includes 4 if-else if statements are expected to change collectively in all 4 parts of the condition, if a change on the guard variable is being made. Regarding specific metrics, it is interesting to highlight the very high dispersion of metrics, since only one metric (LCOM) is found in two studies: all the rest are used only ones.
- Regarding ***change proneness***, we have observed that the most frequently used property is *coupling*. This assertion can be considered reasonable, because coupling is the prevalent quality property for capturing *instability:* instability can be viewed as a consequence of change proneness (Arvanitou et al., 2017b). Second ranks the property *size,* which can be considered intuitive, in the sense that larger artifacts (e.g., classes) are by nature more probable to change in a next version of the system, since they are probably related to more requirements and are probably receiving more ripple effects from other classes (Lu et al., 2012). Similarly to before, in the top-5 metrics for assessing change proneness, we have not identified any metric related to coupling (LoC, CC, DIT, LCOM, and NOC): highlighting again the high number of distinct coupling metric that hinders them from being well-established in the literature.
- Finally, with respect to ***changeability***, we have identified *coupling, inheritance*, and *complexity* as the most prevalent quality properties. On the one hand, coupling is inversely related to changeability, since artifact with high coupling are more rigid and less flexible into changes. On the other hand, the use of inheritance is positively linked to changeability, in the sense that inheritance enables the application of various good practices for extendibility, e.g., design patterns (Gamma et al., 1995), Open-Closed Principle (Martin, 2003), etc. Finally, the existence of complex methods (usually long methods with various selection statements) is an indicator of a difficult to change part of the software.

Table 6
List of most frequently properties for each CIA parameter.

CIA parameters	Quality property	Number of metrics	Pct. of studies
Instability	Coupling	29	71%
	Complexity	7	43%
	Size	10	38%
	Inheritance	2	28%
	Cohesion	4	21%
	Fault-Reliability	4	7%
	Business	2	7%
	Encapsulation	1	7%
	History	1	7%
	Testability	1	7%
Change proneness	Coupling	72	86%
	Size	38	71%
	Cohesion	14	43%
	Inheritance	12	43%
	Complexity	9	43%
	Encapsulation	5	28%
	Polymorphism	7	14%
	History	7	14%
	Comments	3	14%
	Other	2	14%
	Business	2	7%
Amount of change	Size	21	60%
	Inheritance	15	60%
	Cohesion	7	60%
	Complexity	3	60%
	Coupling	14	40%
	Other	2	40%
	Encapsulation	4	20%
Changeability	Coupling	5	66%
	Inheritance	2	33%
	Size	4	33%
	Complexity	9	33%
	Business	1	33%
	History	1	33%

Based on the above we can draw the following conclusions: (a) indirect metrics for CIA parameters assessment are substantially more, compared to direct assessors; (b) the plethora of indirect assessors seems to hinder the establishment of common metrics for the quantification of CIA parameters, leading to confusion among researchers; and (c) despite the aforementioned facts, the relations of metrics and CIA parameters are straightforward and intuitive, suggesting that they are all in the right direction and have merit in practice.

Exploration of CIA at Various Development Phases. Next, in Fig. 6, we present the count of studies that investigate CIA parameters quantification at specific development phase. The results suggest that the most frequently studied development phase is *implementation* (62%), followed by *design* (22%), *architecture* (14%), and *requirements* (2%). Following a similar route of investigation regarding the studied software artifacts, the results show that *source code* is the most frequently studied artifact. This outcome is expected, since it follows the distribution of studies to development phases.

Finally, Table 7 presents the results of cross-tabulating development phases and change impact parameters. In particular, for

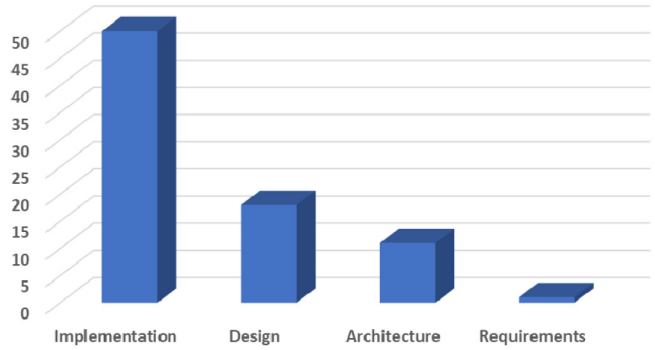

Fig. 6. Frequency of the development phases.

each development phase, we record the corresponding change impact parameter and the number of studies in which they have been explored. Based on Table 7, *instability* is examined in all development phases with a very high score. This finding can be explained by the fact that the assessment of an uncertain phenomenon (i.e., which artifacts are going to be affected by a change in another part of the system) is more wicked, and challenging for research purposes. Additionally, *change proneness* is well studied during implementation and design phase. This can be attributed to the fact that for source code and design artifacts (e.g., number of classes) the frequency of their change can be often directly measured from source code repositories. However, other development phases (e.g., requirements) could not use repositories, in the sense that the evolution of the artifacts (e.g., use cases) do not record from the companies or open source software.

A similar analysis, i.e., cross-tabulating CIA parameters and software artifacts (see Appendix D) has revealed that *source code* is the most studied software artifact in all CIA parameters. This is an expected outcome in the sense that *source code* is available for all software projects, and the majority of the quality assessment concern the source code (Arvanitou et al., 2017a). Furthermore, we have noticed that the order of appearance of the architectural and design artifacts it is not the same in all quality attributes. This could be explained based on the aforementioned rationale (regarding the availability of artifacts for CIA), as well as the fact that the dividing line between architecture and detail-design is often subtle, and could thus, could probably be a misunderstanding between researchers about the boundaries of these two phases.

Based on the aforementioned results, we can claim that CIA parameters exploration by development phase and artifact have led to similar results, which are primarily driven by: (a) the availability of artifacts in each development phase; and (b) curiosity/research challenge, i.e., targeting at the CIA parameter that is the most uncertain one, i.e., instability.

4.3. Research direction in change impact analysis (RQ_3)

In this section, we present the research goals of papers that focus on change impact analysis. Based on the Open-Sorting process (see Section 3.4), we have identified four main themes that were repeating for all change impact parameters (in total 16 themes). The themes concerning instability are outlined below (the rest are omitted, since they are repetitive):

- ***quantify instability based on other metrics***: The authors of primary studies quantify instability using quality metrics (directly or indirectly);
- ***assess the effect of a phenomenon on instability***: The authors of primary studies explore the level of the impact of one specific phenomenon (e.g., code clones) on instability;
- ***assess the effect of instability to other phenomena***: The authors of primary studies investigate if there is an effect of instability to another phenomenon (e.g., on fault proneness);
- ***propose a novel instability metric***: The authors of the primary studies propose an instability metric.

Table 7
List of most frequently CIA parameters for each development phase.

Development phase	CIA parameters	Freq.
Implementation	Change proneness	20
	Amount of change	17
	Instability	15
	Changeability	4
Architecture	Instability	9
	Amount of change	3
	Changeability	1
	Change proneness	1
Design	Change proneness	7
	Instability	5
	Amount of change	3
	Changeability	4
Requirements	Instability	1
	Changeability	1

The frequency of themes' occurrence (per change impact parameter) is presented in Table 8, whereas the full results (as well as examples of each theme) can be found in Appendix E.

In a cumulative perspective (by examining each row of Table 8), the most studied theme is the assessment of the *effect of a phenomenon on CIA parameters* (31%); followed by the *quantification of a CIA parameter* based on other metrics (24%) and the proposal of a *novel CIA parameter metrics or methods* (24%). Finally, the assessment of the *effect of CIA parameter values on other phenomena* is studied on 21% of the studies. Based on the findings of Table 8 and Appendix E, the following observations can be stated:

- ***Relation of CIA to other phenomena***. Despite the fact that quite some papers explore the relation between CIA parameters and other phenomena (and vice-versa), we believe that there is still room for future work in these directions, since only a limited number of pairs between CIA parameters and software engineering phenomena have been studied more than two times (see Appendix E). Therefore, replication studies that would increase the level of evidence on such relations are required. By focusing on the direction of such explorations, we can observe that potential of future research concerning the *instability* and *amount of change* themes present a good balance, whereas for *change proneness* and *changeability* there is a lack of many primary research studies focusing on their effect on other phenomena.
- ***Assessment based on Other Metrics***. Based on the aforementioned distribution of studies we can claim that a plethora of ways to assess CIA parameters based on other metrics exist in practice (as it is demonstrated by the information shown in Appendix C). The over-researching on the identification of more and more metrics correlated to one phenomenon is a common belief among software engineering researchers and practitioners (e.g., for coupling there are more than 30 (Briand et al., 1999)). Therefore, in general, although we acknowledge the need for further research towards fine-tuning of existing approaches aimed to predicting the values

Table 8
Cross-tabulation of research goal themes and CIA parameters.

	Instability	Change proneness	Amount of change	Changeability
Quantification of a CIA parameter based on other metrics	10	14	5	3
Assess the effect of a phenomenon on CIA parameters	9	15	8	6
Assess the effect CIA parameter to other phenomena	14	3	7	0
Propose a novel CIA parameter metric or method	15	6	3	2

of some parameters, we highly encourage researchers to focus their efforts on the improvement of the industrial relevance of their results, rather than attempting to link more metrics to the CIA parameters.

- ***Quantification by Methods vs. Metrics***. On the other hand, the proposal of novel metrics for directly quantifying the CIA parameters lag compared to the aforementioned correlation studies. Based on Appendix E, there are nine (9) novel metrics targeting to *instability* calculation, whereas for *change proneness* and *changeability* there is only one (1) metric. Regarding the *amount of change*, we have not identified any method; its direct quantification is achieved only through metrics. Therefore, we believe that future research is required in this direction.

5. Research roadmap

In this section we present the implications of our study for researchers. We split the discussion on research implications on two parts: First, we present over-studied areas of CIA and explain possible reasons on this. Second, we present a tentative research roadmap, in terms of research areas that deserve further investigation.

Along our analysis, we have identified two over-studied areas. The correlation of existing metrics and CIA parameters; especially focusing on instability and change proneness. This fact can be explained in two ways: first, it seems like a convenience choice in the sense that there is a plethora of available source code metrics, which can be explored in various studies as predictors of CIA parameters. Second, it seems that the over-study of change proneness also seems as an easy target for researchers in the sense that change proneness (change frequency) can be very easily captured by exploiting source code repositories. On the other hand, the over-study of instability seems as a novel target in the sense that it covers an interesting research direction, this of the ripple effects. Studying ripple effects is interesting for researchers, since it is a non-trivial task, which if successfully completed yields substantial improvements for maintenance costs. Nevertheless, despite the interest of this research direction, we believe that the approximation of the phenomenon is saturated, and the problem should be approached differently in future work endeavors.

On the other hand, based on our findings, some areas need additional exploration: (a) proposal of novel metrics and methods for direct quantification of CIA parameters; (b) proposal of a novel metric for change proneness and changeability quantification; (c) provision of empirical evidence on the effect of CIA parameters on other phenomena, and vice-versa for replication purposes; and (d) empirical assessment of the effect of changeability and change proneness on other phenomena. First, given the over-studying of proxying CIA parameters and the lack of specialized metrics, we believe that a research roadmap must suggest researchers to avoid ***empirically exploring the relation of existing metrics to CIA parameters, but encourage them to propose novel, direct, and more accurate indicators***. The lack of such ***indicators*** is more evident for ***changeability*** and ***change proneness***. Second, we suggest that although there is a general belief that CIA is important and shall be performed, there is a lack of evidence on which phenomena are affected by CIA, and which aspects of software development are affected by efficient CIA. Therefore, we ***encourage researchers to seek for rigorous and industrially relevant evidence on the way that CIA is applied and how it is related to other phenomena***. As an example, we believe that a study that monetize the benefits obtained by CIA would be very interesting for practitioners and would strengthen the understanding of managerial stakeholders on the benefits from performing CIA. Such an understanding, would enable the increase of investment in CIA and elevate it as a standard practice in industry; leading to a cultural change that would enable the upfront design for changeability, in a focused way; improving the quality of design hotspots.

6. Threats to validity

In this section we present the threats to validity of the current study based on guidelines for identifying, reporting, and mitigating threats to validity, specialized for secondary research studies in software engineering, as they are suggested by Ampatzoglou et al. (2019).

Study Selection Process. Study selection validity concerns the early phases of the research, i.e., the search process and the filtering of studies. To guarantee that our search process adequately identified all relevant studies (from the studied top-quality venues) we used a well-defined process, based on strict guidelines (Kitchenham and Charters, 2007). To guarantee the relevance to software engineering, the identification process consisted of an automated search of thirteen well-known venues that publish only SE studies. The search string was extensive and constructed in a systematic way (see Section 3.2), in the sense that we have used only the name of the change impact parameters, so to return candidate primary studies that are related to change impact analysis. However, it could be possible to exclude studies that have used different terminology from the more established ones. The benefit of focusing studies that are using standard terminology is that the use of subjective criteria to characterize the change impact parameters has been avoided. To mitigate the threat to miss relevant studies, a quasi-gold standard has been used. More specifically, we have checked that all primary studies of a broader secondary study (i.e., Arvanitou et al., (2018)) published in three venues (namely TSE, IST, and JSS) have been retrieved from applying our search string. Furthermore, in the inclusion/exclusion phase, it could be possible to exclude relevant articles. To mitigate this threat, we used three authors in the selection process, discussing any potential conflicts and a systematic voting procedure. After this process, a four and the fifth authors have randomly screened a subset (15%) of the studies chosen for inclusion to verify the choice, without identifying any problems. Also, the inclusion/exclusion criteria have been extensively discussed by the authors to ensure their clarity and to avoid misinterpretations. Furthermore, from our searching process we have excluded gray literature, since the goal of the study focuses on the use of metrics and methods, which are almost never published in gray literature. Our study is not suffering from missing non-English papers and the papers published in a limited number of journals and conferences, since our search process was aiming at a large number of publication venues all publishing papers only in English. Finally, we were able

to access all publications because our institutions provide access to DLs.

Data Validity. In terms of data validity, the main threat is related to data extraction bias and the selection of specific venues. Concerning the first, all relevant data were extracted and recorded manually by the second and the third author. Due to the potential for subjectivity in this process (e.g., regarding the mapping of artifacts in specific development phases), two authors reviewed and further refined the collected data, re-validating them. After this process, the results were discussed among all authors and they resolved any conflicts. Regarding the decision to limit our search space to specific venues (to ensure the high quality of the studied research corpus), we acknowledge the fact that some data points have been missed. Nevertheless, this decision guarantees (to some extent) the quality of primary studies, and therefore the results of the secondary study. Additionally, there is no publication bias in the selected studies, in the sense that the primary studies have been retrieved by various venues. Thus, the aforementioned studies are not affected by a closed and small circle of researchers. Our mapping study is not affected from the following threats: (a) small sample size, as it became possible to recover 99 articles; (b) lack of relationships, the study did not aim to identify relationships between data, but only to classify and compose; and (c) the selection of variables to be extracted, as the research questions of this study did not create disagreements in the discussions between authors based on the variables to be extracted. Moreover, we did not identify issues with the use of statistical analysis, in the sense that the nature of our research questions did not require hypothesis testing, but only basic statistical analysis (descriptive statistics). Finally, to mitigate the researchers' bias in data interpretation and analysis, the authors discussed the data clustering based on the goals of the primary studies, the change impact parameters, and the research goals that have been used. We note that some explanations express the viewpoints and personal opinion of the authors, based on the understanding of the results.

Research Validity. In terms of research validity, threats are related with research method bias and repeatability. Regarding the first one, the majority of the authors are very familiar with the process of conducting secondary studies, as they have participated in a large number of secondary studies as co-authors and reviewers. On the other hand, it could be argued that the following evaluation process ensures the reliability and replication of this study. Therefore, all important decisions for the review process have been thoroughly documented in this manuscript and can be easily reproduced by other researchers. Second, the fact that the export of data is based on the opinion of three authors can to some extent guarantee the reduction of potential bias. Finally, all extracted data have been made public so that the results can be compared and validated[2]. Additionally, through discussion among the authors, we have defined three main research questions in which they accurately map to the study goal. This is clearly illustrated by the mapping of each research question to the research objectives/goals. Furthermore, in the literature we have been able to identify a substantial amount of related works that can be used for comparison to our results. In particular, for this reason we used related studies for performing change impact analysis and maintainability predictors. Finally, the selection of the research method is adequate for the goal of this study and no deviations from the guidelines have been performed.

7. Conclusions—Implications for practitioners

In this paper we presented the outcomes of a systematic mapping study on Change Impact Analysis targeting at three distinct goals: (a) explore the practical benefits of change impact analysis; (b) provide an overview of CIA parameters (instability, change proneness, amount of change, and changeability) quantification metrics and methods; and (c) characterize the research landscape as over- or under-researched. To achieve these goals, we explored more than 500 articles, out of which we proceeded at data extraction on 99. Regarding the first goal, we have provided evidence that all maintenance activities (i.e., addition of new features, bug fixing, performance of quality improvements) can benefit from change impact analysis. This finding confirms the industrial relevance of CIA, since clear benefits to practitioners are demonstrated, and the link between academia and practice is highlighted. With respect to quantification of change impact parameters, our results suggest that the research community has already validated the relation between change impact parameters and a vast amount of metrics. However, the enormous number of metrics can to some extent cause confusion, since there is not enough evidence for specific metrics and the practitioners have to deal with a complicated metric selection process. On the other hand, the direct quantification of the change impact parameters with specialized metrics appears to lag: in the cases of change proneness, we have identified zero papers that propose novel metrics for its quantification. Nevertheless, the majority of proposed metrics appear to have an intuitive relation to CIA parameters, providing a hint for their validity. Finally, with respect to research directions, we believe that most of future work emphasis should be placed on understanding the effect of CIA on phenomena, and vice-versa.

Concluding, we encourage practitioners to perform change impact analysis (regardless of the type of change), so as to reduce the maintenance effort and the number of introduced bugs while applying the change. To support effective CIA, practitioners should first identify change prone artifacts and artifacts that usually attract high volumes of change amounts. For these artifacts, the practitioners must take specific measures to improve their changeability. Such a precautionary action is expected to reduce maintenance costs, in the design hotspots, i.e., parts of the system that change regularly and largely. Next, upon the application of the change, the practitioners should assess the instability of artifacts (due to the ripple effect)—based on class dependencies. For the highly instable artifacts, additional testing must be performed. Based on our findings, the aforementioned assessments can be performed with sophisticated metrics or methods, which produce accurate CIA, but also with proxies such as coupling and size metrics. Therefore, based on the level of investment of a company on CIA (probably a function of the maintenance costs), the company can either use specialized tools, or rely on more generic metric calculation tools.

CRediT authorship contribution statement

Maria Kretsou: Conceptualization, Methodology, Formal analysis, Data curation. **Elvira-Maria Arvanitou:** Conceptualization, Methodology, Formal analysis, Data curation, Writing - original draft, Writing - review & editing. **Apostolos Ampatzoglou:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Ignatios Deligiannis:** Review & editing. **Vassilis C. Gerogiannis:** Review & editing.

Declaration of competing interest

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

Acknowledgment

Work reported in this paper was financially supported by the action "Strengthening Human Resources Research Potential via Doctorate Research" of the Operational Program "Human Resources Development Program, Education and Lifelong Learning, 2014-2020", implemented from State Scholarship Foundation (IKY) and co-financed by the European Social Fund and the Greek public (National Strategic Reference Framework (NSRF) 2014–2020).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jss.2020.110892.

References

Alam, K.A., Ahmad, R., Akhunzada, A., Md Nasir, M.H.N., Khan, S.U., 2015a. Impact analysis and change propagation in service-oriented enterprises: A systematic review. Inf. Syst. 54, 43–73.

Alam, K., Ahmad, R., Akhunzada, A., Nasir, M., Khan, S., 2015b. Impact analysis and change propagation in service-oriented enterprises: A systematic review. Inf. Syst. 54, 43–73.

Aljohani, Qureshi, 2016. Management of changes in software requirements during development phases. Int. J. Educ. Manag. Eng. 6 (6), 12–26.

Alshayeb, M., Li, W., 2005. An empirical study of system design instability metric and design evolution in an agile software process. J. Syst. Softw. 74, 269–274.

Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., Chatzigeorgiou, A., 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Inf. Softw. Technol. 106.

Ampatzoglou, A., Stamelos, I., Gkortzis, A., Deligiannis, I., 2012. A methodology on extracting reusable software candidate components from open source games. In: 16th International Academic MindTrek Conference (MindTrek '12). Association for Computing Machinery, New York, NY, USA, pp. 93–100.

Arisholm, E., 2006. Empirical assessment of the impact of structural properties on the changeability of object-oriented software. Inf. Softw. Technol. 48 (11), 1046–1055.

Arisholm, E., Sjøberg, D.I.K., Jørgensen, M., 2001. Assessing the changeability of two object-oriented design alternatives a controlled experiment. Empir. Softw. Eng. 233–271.

Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., 2015. Introducing a ripple effect measure: a theoretical and empirical validation. In: 9th International Symposium on Empirical Software Engineering and Measurement (ESEM' 15). IEEE Computer Society, China.

Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., 2017a. A method for assessing class change proneness. In: 21st International Conference on Evaluation and Assessment in Software Engineering (EASE '17). ACM, Sweden.

Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Galster, M., Avgeriou, P., 2017b. A mapping study on design-time quality attributes and metrics. J. Syst. Softw. 127 (5), 52–77, Elsevier.

Balogh, G., Antal, G., Beszédes, A., Vidács, L., Gyimóthy, T., Végh, Á.Z., 2015. Identifying wasted effort in the field via developer interaction data. In: International Conference on Software Maintenance and Evolution (ICSME), Bremen, pp. 391-400.

Basili, V.R., Caldiera, G., Rombach, H.D., 1994. Goal Question Metric Paradigm, Encyclopedia of Software Engineering. John Wiley & Sons, pp. 528–532.

Benestad, H.C., Anda, B., Arisholm, E., 2009. Understanding software maintenance and evolution by analyzing individual changes: a literature review. J. Softw. Maint. Evol.: Res. Pract. 21, 349–378.

Boehm, B.W., 1991. Software risk management: Principles and practices. IEEE Softw. 1991, 32–42.

Bohner, S.A., 2000. Impact analysis in the software change process: A year 2000 perspective. In: 4th International Conference on Software Maintenance (ICSM' 96). IEEE Computer Society, Monterey, USA, pp. 42–51.

Briand, L., Daly, J., Wust, J., 1999. A unified framework for coupling measurement in object-oriented systems 25 (1) 91-121.

Brink, C., Heisig, P., Wackermann, F., 2016. Change impact in product lines: A systematic mapping study. In: International Conference on Information and Software Technologies. pp. 677–694.

Brown, N., Nord, R.L., Ozkaya, I., Pais, M., 2011. Analysis and management of architectural dependencies in iterative release planning. In: Ninth Working IEEE/IFIP Conference on Software Architecture, Boulder, CO, pp. 103-112.

Cai, K.Y., Card, David, 2008. David card an analysis of research topics in software engineering – 2006. J. Syst. Softw. 81 (6), 1051–1058.

Chen, J.-C., Huang, S.-J., 2009. An empirical analysis of the impact of software development problem factors on software maintainability. J. Syst. Softw. 82 (6), 981–992.

Chidamber, S.R., Kemerer, C.F., 1994. A metrics suite for object oriented design. Trans. Softw. Eng. 20 (6), 476–493, IEEE Computer Society.

D'Amorim, F., Borba, P., 2012. Modularity analysis of use case implementations. J. Syst. Softw. 85, 1012–1027.

Decan, A., Mens, T., Grosjean, P., 2019. An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empir. Softw. Eng. 24, 381–416.

Farhoodi, R., Garousi, V., Pfahl, D., Sillito, J., 2013. Development of scientific software: A systematic mapping, a bibliometrics study, and a paper repository. Int. J. Softw. Eng. Knowl. Eng. 23 (4).

Galorath, D.D., 2008. Software total ownership costs: development is only job one. Softw. Tech News 11 (3).

Galster, M., Weyns, D., Tofan, D., Michalik, B., Avgeriou, P., 2014. Variability in software systems - a systematic literature review. IEEE Trans. Softw. Eng. 40 (3), 282–306.

Gamma, E., Helms, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, Reading, MA.

Haney, F.M., 1972. Module Connection Analysis: A Tool for Scheduling of Software Debugging Activities, Fall Joint Computer Conference. IEEE Computer Society, Anaheim, USA, pp. 173–179.

Harrison, R., Counsell, S.J., Nithi, R.V., 1998. An evaluation of the MOOD set of object-oriented software metrics. Trans. Softw. Eng. 24 (6), 491–496.

Hindle, A., 2015. Green mining: a methodology of relating software change and configuration to power consumption. Empir. Softw. Eng. 20, 374–409.

ISO/IEC 9126-1:2001, 2001. Software engineering - Product quality (Part 1: Quality model), Geneva, Switzerland, 2001.

Ivarsson, M., Gorschek, T., 2011. A method for evaluating rigor and industrial relevance of technology evaluations. Empir. Softw. Eng. 16, 365–395, Springer.

Jaafar, F., Guéhéneuc, Y.-G., Hammel, S., Antoniol, G., 2014. Detecting asynchrony and dephase change patterns by mining software repositories. J. Softw.: Evol. Process. 26 (1), Wiley & Sons.

Jabangwe, R., Börstler, J., Šmite, D., Wohlin, C., 2014. Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review. Empir. Softw. Eng. 20 (3), 640–693, Springer.

Kabaili, H., Keller, R.K., Lustman, F., 2005. Assessing object-oriented software changeability with design metrics. In: IASTED International Conference on Software Engineering.

Karanatsiou, D., Li, Y., Arvanitou, E.M., Misirlis, N., Wong, W.E., 2019. A bibliometric assessment of software engineering scholars and institutions (2010–2017). J. Syst. Softw. 147, 246–261.

Khomh, F., Gueheneuc, Y., Antoniol, G., 2009. Playing roles in design patterns: An empirical descriptive and analytic study. In: 2009 IEEE International Conference on Software Maintenance, Edmonton, AB, pp. 83-92.

Khomh, F., Penta, M.D., Guéhéneuc, Y., et al., 2012. An exploratory study of the impact of antipatterns on class change- and fault-proneness. Empir. Softw. Eng. 17, 243–275.

Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009a. Systematic literature reviews in software engineering – a systematic literature review. Inf. Softw. Technol. 51 (1), 7–15, Elsevier.

Kitchenham, B., Brereton, P., Turner, M., Niazi, M., Linkman, S., Pretorius, R., Budgen, D., 2009b. The impact of limited search procedures for systematic literature reviews a participant-observer case study. In: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM'09). IEEE Computer Society, USA.

Kitchenham, B., Charters, S., 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University.

Kosti, M.V., Georgiadis, K., Adamos, D.A., Laskaris, N., Spinellis, D., Angelis, L., 2018. Towards an affordable brain computer interface for the assessment of programmers' mental workload. Int. J. Hum.-Comput. Stud. 115, 52–66.

van Koten, C., Gray, A., 2006. An application of Bayesian network for predicting object-oriented software maintainability. Inf. Softw. Technol. 48 (1), 59–67.

Li, W., Henry, S., "Object-oriented metrics that predict maintainability", (23:2), 1993, pp. 111-122.

Li, B., Sun, X., Hareton, L., Zhang, S., 2013. A survey of code-based change impact analysis techniques.. J. Softw.: Test. Verif. Reliab. 23 (8), 613–646, Wiley.

Lippert, M., Roock, S., 2006. Refactoring in Large Software Projects, first ed. Wiley & Sons.

Lu, H., Zhou, Y., Xu, B., Leung, H., Chen, L., 2012. The ability of object-oriented metrics to predict change-proneness: a meta-analysis. Empir. Softw. Eng. 17 (3), 200–242, Springer.

Malhotra, R., Bansal, A.J., 2016. Software change prediction: a literature review. Int. J. Comput. Appl. Technol. 54 (4), 240–256, 2016.

Malhotra, R., Chug, A., 2016. Software maintainability: Systematic literature review and current trends. Int. J. Softw. Eng. Knowl. Eng. 26 (8), 1221–1253.
Malhotra, R., Khanna, M., 2019. Software change prediction: A systematic review and future guidelines. e-Inform. Softw. Eng. J. 13 (1), 227–259.
de Marco, T., 1986. Controlling software projects: Management, measurement, and estimates.
Martin, R.C., 2003. Agile Software Development: Principles, Patterns and Practices. Prentice Hall.
Misirli, T., A, Shihab, E., Kamei, Y., 2016. Studying high impact fix-inducing changes. Empir. Softw. Eng. 21, 605–641.
Mondal, M., Rahman, M.S., Roy, C.K., Kevin, A., 2018. Is cloned code really stable?. Empir. Softw. Eng. 23, 693–770.
Moser, R., Pedrycz, W., Succi, G., 2004. Analysis of the Reliability of a Subset of Change Metrics for Defect Prediction, Empirical Software Engineering Measurement, 1-4, Conference, City, State, Country.
Palomba, F., Panichella, A., Zaidman, A., Oliveto, R., De Lucia, A., 2018. The scent of a smell: An extensive comparison between textual and structural smells. Trans. Softw. Eng. 44 (10), 977–1000.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: *12th* International Conference on Evaluation and Assessment in Software Engineering (EASE'08). British Computer Society Swinton, Bari, Italy, pp. 68–77.
Power, J.F., Malloy, B.A., 2004. A metrics suite for grammar-based software. J. Softw. Maint. Evol.: Res. Pract. 16 (6).
Riaz, M., Mendes, E., Tempero, E., 2009. A systematic review on software maintainability prediction and metrics. In: *3rd* International Symposium on Empirical Software Engineering and Measurement (ESEM'09). IEEE Computer Society, Florida, USA, pp. 367–377.
Rovegard, P., Angelis, L., Wohlin, C., 2008. An empirical study on views of importance of change impact analysis issues. Trans. Softw. Eng. 34 (4), 516–530, IEEE Computer Society.
Saraiva, J., Barreiros, E., Almeida, A., Lima, F., Alencar, A., Lima, G., Soares, S., Castor, F., 2012. Aspect-oriented software maintenance metrics: A systematic mapping study. In: *16th* International Conference on Evaluation & Assessment in Software Engineering (EASE'12). IEEE Computer Society, Ciudad Real, Spain, pp. 253–262.
Shaheen, M.R., Bousquet, L.d., 2009. Is depth of inheritance tree a good cost prediction for branch coverage testing?. In: First International Conference on Advances in System Testing and Validation Lifecycle, Porto, pp. 42-47.
Spencer, D., 2009. Card Sorting: Designing Usable Categories, first ed. Rosenfeld Media.
Stevanetic, S., Zdun, U., 2018. Supporting the analyzability of architectural component models - empirical findings and tool support. Empir. Softw. Eng. 23, 3578–3625.
Sun, X., Leung, H., Bin, L., Bixin, L., 2014. Change impact analysis and changeability assessment for a change proposal: An empirical study. J. Syst. Softw. 96, 51–60.
Tizzei, P.L., Dias, M., Rubira, C.M.F., Garcia, A., Jaejoon, L., 2011. Components meet aspects: Assessing design stability of a software product line. Inf. Softw. Technol. 53, 121–136.
van Vliet, H., 1993. Software Engineering: Principles and Practice, third ed. Wiley, Chichester, England.
Woo, G., Chae, H., S, Cui, F.J., Ji, J.-H., 2009. Revising cohesion measures by considering the impact of write interactions between class members. Inf. Softw. Technol. 51, 405–417.
Yau, S.S., Collofello, J.S., MacGregor, T.M., 1978. Ripple effect analysis of software maintenance. In: *2nd* International Computer Software and Applications Conference (COMPSAC' 78). IEEE Computer Society, pp. 60–65.
Zhou, Y., Leung, H., 2007. Predicting object-oriented software maintainability using multivariate adaptive regression splines. J. Syst. Softw. 80 (8), 1349–1361, Elsevier.
Zhou, Y., Xu, B., 2008. Predicting the maintainability of open source software using design metrics. Wuhan Univ. J. Nat. Sci. 13 (1), 14–20, Springer.

Maria Kretsou is an M.Sc. student at the Department of Informatics in the Open Hellenic University. She holds a B.Sc. degree in Physics from the Aristotle University of Thessaloniki, Greece (2010). Currently she works in public services as a civil servant. Her research interests include software metrics, software analysis and design, and software maintenance.

Dr. Elvira-Maria Arvanitou is a Post-Doctoral Researcher at the Department of Applied Informatics, in the University of Macedonia, Greece. She holds a Ph.D. degree in Software Engineering from the University of Groningen (Netherlands, 2018), an M.Sc. degree in Information Systems from the Aristotle University of Thessaloniki, Greece (2013), and a B.Sc. degree in Information Technology from the Technological Institute of Thessaloniki, Greece (2011). Her Ph.D. thesis has been awarded as being part of the top-3 ICT-related in Netherlands for 2018. Her research interests include technical debt management, software quality metrics, and software maintainability.

Dr. Apostolos Ampatzoglou is as an Assistant Professor in the Department of Applied Informatics in University of Macedonia (Greece), where he carries out research in the area of software engineering. Before joining University of Macedonia he was an Assistant Professor in the University of Groningen (Netherlands). He holds a B.Sc. on Information Systems (2003), an M.Sc. on Computer Systems (2005) and a Ph.D. in Software Engineering by the Aristotle University of Thessaloniki (2012). He has published more than 100 articles in international journals and conferences, and is/was involved in over 15 R&D ICT projects, with funding from national and international organizations. Also, he has been nominated as the 3rd most active Early-Stage Researcher in software engineering for the period 2010–2017. His current research interests are focused on technical debt management, software maintainability, game engineering, software quality management, open source software, and software design.

Dr. Ignatios Deligiannis is Professor at International Hellenic University. He was member of ESERG (Empirical Software Engineering Research Group at Bournemouth University, UK). He received his B.Sc. in computer science from the University of Lund, Sweden, then worked for several years in software development at Siemens Telecommunications industry. His main interests are object-oriented software assessment, and in particular design heuristics and measurement.

Dr. Vassilis C. Gerogiannis holds a Diploma in Computer/Software Engineering and a Ph.D. in Software Engineering from the University of Patras, Greece. He is a full time Professor in the Department of Digital Systems at the University of Thessaly, Greece. He is also Adjunct Professor, teaching Software Engineering, at the Hellenic Open University. From 1992 until present, he has participated as software engineer, project manager and research director in several R&D projects funded by EU or national organizations. He has authored/co-authored more than 120 papers published in international journals/conference proceedings, which have been cited in a plethora of citations. He acts as guest editor, member of the editorial board and reviewer in international journals. He serves as program chair, member of the organization/technical committee and invited speaker in international conferences. He has received the "best paper award" in two international conferences. His research interests include Software Engineering, Project Management and Decision Making.