Recent Trends in Software Quality Interrelationships: A Systematic Mapping Study

Michael Y. Shoga*, Celia Chen†, Barry Boehm*
*Center for Systems and Software Engineering, University of Southern California, Los Angeles, USA
Email: {mshoga, boehm}@usc.edu
†Department of Computer Science, Occidental College, Los Angeles, CA
Email: {qchen2}@oxy.edu

***Abstract*—Despite the importance of software qualities, they are not well understood, especially in the context of the interrelationships between qualities. A number of systematic mapping studies have been conducted prior to 2015 to summarize the literature on the topic and to identify research gaps. To provide a better understanding of the current state of the art, we conducted a systematic mapping study on relevant studies from 2015 to 2019 through a database search and a subsequent snowballing approach. In total, 18 studies were selected as the study subjects wherein we evaluated the types of software quality interrelationships and the qualities that comprise them. Based on our findings, we report on the progress made to address previously identified research gaps.**

***Index Terms*—software quality, software quality interrelationships, systematic mapping study**

I. Introduction

Software qualities play a critical role in the potential for success of software systems. The realization of these qualities and the non-functional requirements that specify them often depend on decisions made across the system and can have a multiplicative effect on the overall cost of the system [1]. Despite their significance, they are considered to be poorly defined, understood, and difficult to measure [2]. A compounding difficulty that arises when addressing non-functional requirements lies in the relationships between the quality attributes. These relationships are apparent, as making changes to a system to address one software quality can affect one or more other qualities. Thus one often needs to prioritize and balance between the various software qualities depending on the system and its stakeholders.

Various models and frameworks have been proposed to characterize the relationships between qualities, and there has been ongoing work to attempt to address conflicts between qualities [3] [4] [5]. However, previous studies and reviews looked at conflicts between software qualities and have identified gaps in the literature:

1) Some studies did not provide any definitions for the quality attributes [2] [6] [7] [8].
2) A very wide range of vocabulary was found to describe software quality attributes [2] [6] [7] [8] [9].
3) A very small amount of research provided any empirically validated results [7] [8] [9] [10].
4) There was a lack of discussion of research validity on the approaches [9].
5) Quality trade-off approaches predominantly focused on earlier phases of software development [9].
6) Automatic approaches are needed in quality interrelationships detection [10] [11].
7) The nature of these quality interrelationships were still not captured [7] [8].

In the interest of providing a better understanding of the current state of the art in this field and evaluating the progress in addressing those research gaps identified in previous works, we have conducted a systematic mapping study in the area of interrelationships between software qualities on studies that were published from 2015 to 2019.

This material is based upon work supported in part by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

II. Related Work

Various related studies and reviews were available in literature prior to 2015 which identified gaps in the software quality interrelationship space.

Svennson *et. al* [11], included 18 studies in a literature review study on quality requirements management. The author classified the studies into the following categories: elicitation, dependencies, metrics, cost estimations, and prioritization. However, the authors were not able to identify which method was most suitable to use for identification of interdependencies among quality requirements.

In and Boehm [12], conducted a comparative analysis through a case study on 12 projects using two exploratory knowledge-based tools (QARCC and S-COST) for conflict identification and resolution among stakeholders. Theoretically, QARCC could identify potential quality-conflict issues by generating relationships of the input quality with all the other qualities. An example given was a potential conflict between dependability and interoperability.

García-Mireles *et. al* [6], conducted a review study on a set of empirical publications reported in a mapping study [9]. The authors reported the popular methods used by software organizations to manage interactions between quality characteristics and the types of quality models used to define quality characteristics.

Barney *et. al* [9], provided an overview of software quality trade-offs in general through a systematic mapping study of 168 publications. The study focused on the process of handling quality trade-offs, rather than identifying their nature.

Mairiza *et. al* presented a series of studies on non-functional requirements and the conflicts among them. The authors identified techniques and methods used to manage conflicts between non-functional requirements [8], categorized non-functional requirements into 3 different classifications: definitions; types; and applicable domains [2], and presented potential conflicts among these non-functional requirements [7] [13].

The following review and mapping studies were conducted after 2015; however, they do not fully address software quality interrelationships as described in the following ways.

Aldekhail *et. al* [10], explored approaches of conflict analysis identification through a comparative study including 20 studies dated from 2001 to 2014. The authors classified these studies into 4 classifications, the type of requirements and the type, scope, and representation of the approaches. This comparative study only provided an overview on functional and non-functional requirement conflicts identification, rather than the nature of conflicts themselves.

Me *et. al* [14], performed a systematic literature review on interactions between architectural patterns and quality attributes. They identified several ways to characterize these interactions; however, they note that most of their 99 selected primary studies provided insufficient information to be classified. They also identified a need to further validate the consistency of interactions when the pattern is implemented on its own or in combination with other patterns. While most frequently occurring quality attributes are identified, interrelationships between them are not considered.

Ijaz *et. al* [5], conducted a systematic literature review on non-functional requirement prioritization techniques which included 30 studies from 2008 onward. They identified 25 prioritization techniques as well as common limitations in the approaches. These limitations include a lack of scalability for larger data sets and a lack of validation for some approaches. While interdependencies between functional and nonfunctional requirements are mentioned as a challenge, these interdependencies are not within the scope of the study.

García-Mireles *et. al* [15] investigated the relationships between software product quality characteristics and green in software sustainability (producing more sustainable software products). Based on the primary studies, the authors identified several potential positive and negative interactions between product qualities from ISO/IEC 25010 and sustainability aspects. While this mapping study addresses relationships between quality characteristics and sustainability, it does not speak to the more general interrelationships between the quality characteristics themselves.

III. Research Methodology

This section presents a detailed description of the research methodology of this systematic mapping study.

With the goal of minimizing research bias while maximizing the relevant work to be included, we developed a structured approach guided by the updated guidelines provided by Petersen *et. al* [16]. Our approach consisted of a set of research questions, a search strategy with selection criteria, data extraction, and data analysis.

A. Research Questions

Motivated by the findings from previous systematic studies, we look to assess the trends in software quality interrelationships. Specifically, we answer the following research questions related to the gaps introduced in Section I:

- ***RQ1:*** Are software qualities defined and are they consistent in their definitions?
  This research question looks to address gaps 1 and 2. To answer this research question we investigated the software qualities being mentioned in software quality interrelationships to see whether definitions for quality attributes continue to be omitted in the literature. We further look at the top 5 most commonly considered qualities to examine whether the highly investigated qualities in this area have changed and if the bases for their definitions are consistent across the studies.
- ***RQ2:*** How are software quality interrelationships studies being evaluated?
  This research question looks to address gaps 3 and 4. We first classified each study as to whether they were reporting a relationship or whether they were proposing a method to identify or manage quality interrelationships. We then mapped the studies based on whether they reported an empirical study, a case study, or no evaluation with regards to their methods or interrelationship reporting.
- ***RQ3:*** What approaches are being used in quality interrelationship studies?
  This research question looks to address gaps 5 and 6. Previous work had focused on addressing software quality interrelationships at earlier stages of software development [9], and it was noted that automatic approaches are needed in requirement conflict detection [10]. In our study, we mapped each study as "early" if their methods or interrelationships were applied to process, architecture, or requirements. Approaches were classified as automatic if some tool was used to analyze and detect requirement conflicts as in [10]; we similarly evaluated whether the studies involved use of a tool to address quality interrelationships.
- ***RQ4:*** What are the characteristics of software quality interrelationships?
  This research question looks to address gap 7. Prior work noted that conflict identification techniques were limited to only a high level [8], thus we look to see if this holds for other quality interrelationships. To answer this RQ, we evaluated the types of interrelationships being reported in the studies, as well as whether the studies mentioned a type of metric used to evaluate the strength

of the interrelationship. For this study, interrelationship strength is defined as any description beyond the presence or absence of a quality interrelationship.

B. Database Search

With the goal of obtaining a list of candidate studies, we performed a thorough database search.

1) Search Terms: Taking **PICOC** (Population, Intervention, Comparison, Outcomes, and Context), [17] into consideration, we derived a set of keywords for our research questions (as shown in Table I). Quotes are used for key phrases and wildcards(*) for keywords whose endings transform when pluralized. Since previous work indicated that many studies were not empirically validated [7] [8] [9] [10], we excluded "Outcome" to ensure all relevant studies were considered.

Related terms within population and intervention were identified from ISO/IEC 24765 (Systems and Software Engineering - Vocabulary) [18] and the Software Engineering Book of Knowledge [19]. To construct the final query, intra-set terms were connected with boolean ORs, and sets were combined with boolean ANDs.

2) Database Sources: IEEEXplore, ACM Digital Library, and Scopus were selected based on the recommendation of the systematic mapping study guidelines to ensure a thorough sampling [16]. The following query was used for title, abstract, and keywords:

(("software engineering") AND ("nonfunctional requirement"OR "quality requirement"OR "design constraint"OR "software qualit* ") AND (relationship OR dependenc* OR conflict OR synerg* OR tradeoff OR interaction)) [1]

C. Study selection

This section describes the selection process of the candidate studies in depth. Figure 1 illustrates the overall process.

1) Selection Criteria: To answer our research questions, we developed a set of inclusion and exclusion criteria to find relevant works.

Inclusion Criteria:

- *Relevance:* Papers should address interrelationships between software product qualities in software engineering.
- *Language:* Papers should be included only if they were available in English.
- *Time period:* Papers should be included if they were from the time span of January 2015 to December 2019.

Exclusion Criteria:

- *Unrelated Software Qualities:* For example, software quality can include process quality, which was considered out of scope.
- *Clarity of relationship:* Papers should be removed if their relevance or subject matter were not clear from the full paper reading.
- *Availability:* Papers should be removed if they were not available in full-text from available databases.

[1]Since ACM Digital Library does not take wild cards into consideration with phrases and does not have a field for keywords, qualit* was substituted with "quality OR software qualities"and search was done for title and abstract.

2) Identification of the start set: The start set can be identified by using the search query described in Section III-B1 to query the online databases mentioned in Section III-B2.

Figure 1 illustrates the overall selection process of our systematic mapping study. Out of 1710 papers from an initial database search, the results were refined by year, 2015-2019, and duplicates were removed to yield 465 papers.

Two rounds of review were conducted on these papers, with consensus meetings held after each round. In the first round, the papers were independently evaluated based on the titles and abstracts. Papers that were deemed relevant by at least one author were passed to continue into the next round, yielding 62 papers. In the second round, the papers were read in full and detailed discussions were followed up. Extended and revised journal versions of conference papers were retained and the conference versions removed from the set.

At the end of this process, we identified 12 relevant papers, which were included in the start set.

D. Snowballing Search

For each paper in the start set, we conducted a backward snowballing search on its references and a forward snowballing search on its citations, following the guidelines in [20].

1) Backward Snowballing: We first refined the year of each referenced paper to exclude any papers that were published before 2015. Then the titles and abstracts were reviewed to determine which to pass into the next round. Next, the inclusion and exclusion criteria were applied based on a full-text reading. Papers that were identified as relevant were added to the start set, and this process was repeated until no new papers were identified. As a result, 530 papers were extracted and 4 papers were added to the start set. At the end of this process, there were 16 papers in the start set.

2) Forward Snowballing: The citations of each paper in the start set were identified from their host site (IEEEXplore, Springer Link, etc.). The same review process was conducted in this search. Relevant papers were added to the starting set, and this process was repeated until no new papers were identified. As a result, 50 papers were examined and 2 papers were added to the start set.

At the end of the forward snowballing search, we obtained 18 relevant papers as the final set for our study.

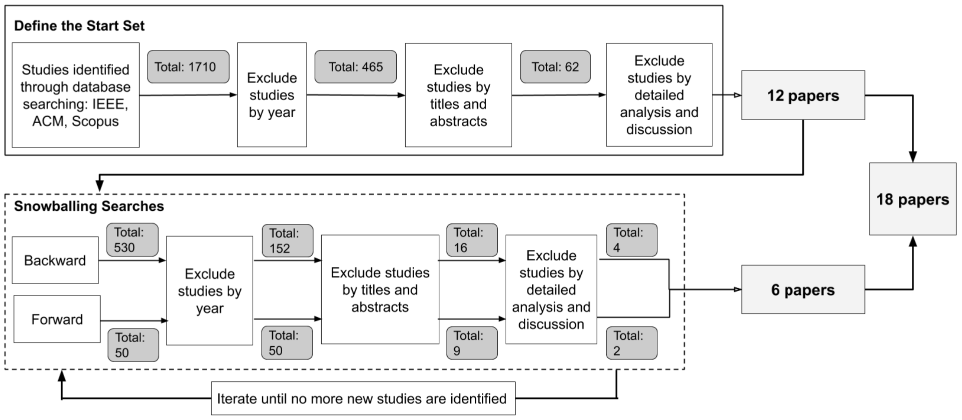
E. Data Extraction and Synthesis

Table II lists the bibliographic and research question data that was extracted from the 18 selected papers and compiled for further analysis.

There were in total 9 conference papers, 6 workshop papers and 3 journal papers included in this study. The year with the most number of publications was 2016 with 5 papers published. While none of the journal or conferences made up a majority of the publication venues, the International Conference on Software Quality, Reliability, and Security (QRS) [21] [22], the International Conference on Software Engineering (ICSE) [23] [24], and the International Requirements Engineering Conference (RE) [25] [26] each had 2 publications within our study set.

TABLE I: PICOC Criteria to Define the Search String for Database Search

	Population	Intervention	Comparison	Outcome	Context
Keywords	software quality	quality interrelationships	not applicable	excluded	software engineering
	"nonfunctional requirement", "quality requirement", "design constraint" "software qualit"*	*relationship, dependenc*, conflict, synerg*, tradeoff, interaction*			



Define the Start Set
Studies identified through database searching: IEEE, ACM, Scopus
Total: 1710
Exclude studies by year
Total: 465
Exclude studies by titles and abstracts
Total: 62
Exclude studies by detailed analysis and discussion
12 papers
18 papers
Snowballing Searches
Backward
Forward
Total: 530
Total: 50
Exclude studies by year
Total: 152
Total: 50
Exclude studies by titles and abstracts
Total: 16
Total: 9
Exclude studies by detailed analysis and discussion
Total: 4
Total: 2
6 papers
Iterate until no more new studies are identified

Fig. 1: The Overall Process of Study Selection

F. *Validity Evaluation*

In this section, the threats to validity in this study are discussed. We consider the following types of validity described in [16]:

1) Descriptive Validity: In this mapping study, this threat can be considered from two perspectives. One is the potential for incorrect assessment of the research publications. As reviewers, we are limited to interpret the intentions of authors. This is also related to the second threat source, the quality of self-reporting of the research publications.

Some of the studies did not have clear objectives, contributions and research methodology. This made the selection process difficult, thus increasing the possibility of inaccurate selection of relevant studies. Moreover, some of the studies used different terminology to describe the same concepts, which could potentially lead to misunderstanding.

To mitigate these threats, we developed a research protocol, which involved independent evaluation of studies and consensus meetings to verify the assessments.

2) Theoretical Validity: The studies collected through the database search may not have adequately covered the literature in the targeted research area due to insufficiency in the search query. To mitigate this threat, we conducted a backward and forward snowballing approach to capture relevant studies which would have been initially missed.

3) Generalizability: Our results may not apply to qualities in other domains that are not within the context of software engineering. The study focuses on software product quality, rather than other types of software quality such as process quality and service quality. Thus, the findings may not apply beyond product quality.

TABLE II: Data Extraction Form

	Data
Bibliographic	Reference Number, Year, Venue Type
RQ 1	Qualities, Quality Basis
RQ 2	Study Purpose, Evaluation
RQ 3	Software Development Stage, Automated
RQ 4	Interrelationships, Interrelationship Strength

IV. Results

This section presents the mapping results and answers the research questions. Table III lists the characteristics of the selected studies in detail.

A. *RQ1*

Overall, 71 distinct qualities were extracted from the selected studies. Figure 2 illustrates the number of qualities extracted per study, including repeated qualities. The most frequently mentioned qualities and the bases for their definitions are compiled in Table IV.

According to Mairiza *et. al* [2], the most commonly considered non-functional requirements were performance, reliability, usability, security, and maintainability. The top five most commonly considered quality attributes extracted from our study were usability, maintainability, security, portability and reliability.

1) Quality definitions: Most of the selected studies mentioned some sort of quality attributes, whether in the format of product quality or non-functional requirements. Out of 18 studies, 7 studies provided clear definitions on the specific qualities mentioned in their research. Moreover, 8 studies gave partial definitions of the quality attributes. These definitions

TABLE III: The Characteristics of the Selected Studies

Study ID	Paper References	RQ 2-1: Study Purpose		RQ 2-2: Evaluation			RQ 3: Software Development Stage		RQ 4: Interrelationship Strength	
		Method	*Reporting*	*Empirical Study*	*Case Study*	*No Evaluation*	*Early*	*Others*	*Metric Reported*	*No Metric Reported*
S1	*[21]*	X		X				X		X
S2	*[27]*	X			X		X		X	
S3	*[23]*	X		X				X	X	
S4	*[28]*	X			X		X		X	
S5	*[29]*	X			X			X	X	
S6	*[30]*		X		X			X		X
S7	*[31]*		X			X	X			X
S8	*[32]*		X		X		X			X
S9	*[24]*		X			X		X		X
S10	*[33]*		X			X		X		X
S11	*[34]*		X			X		X		X
S12	*[35]*		X		X		X		X	
S13	*[1]*		X			X		X		X
S14	*[36]*		X		X		X			X
S15	*[37]*		X			X		X	X	
S16	*[25]*	X			X		X			X
S17	*[22]*		X	X				X		X
S18	*[26]*		X			X	X		X	

TABLE IV: Most Frequent Qualities in Selected Studies

Quality Attribute	Definition Basis	Study ID
Usability	No definition	S6, S14
	ISO/IEC 25010	S2, S7, S11, S15, S18
	SQuaRE series	S3
	ISO/IEC 9126	S10
	Self-defined	S16
Maintainability	No definition	S14
	ISO/IEC 25010	S2, S7, S11, S15
	SQuaRE series	S3
	ISO/IEC 9126	S10
	Self-defined	S17
	The System Qualities Ontology, Tradespace, and Affordability (SQOTA) ontology	S9, S13
Security	No definition	S14
	ISO/IEC 25010	S2, S4, S7, S11, S15, S18
	SQuaRE series	S3
	Self-defined	S16
Portability	ISO/IEC 25010	S7, S11, S15
	SQuaRE series	S3
	ISO/IEC 9126	S10
	The System Qualities Ontology, Tradespace, and Affordability (SQOTA) ontology	S9
	Self-defined	S8, S17
Reliability	ISO/IEC 25010	S2, S7, S11, S15, S18
	SQuaRE series	S3, S5
	ISO/IEC 9126	S10

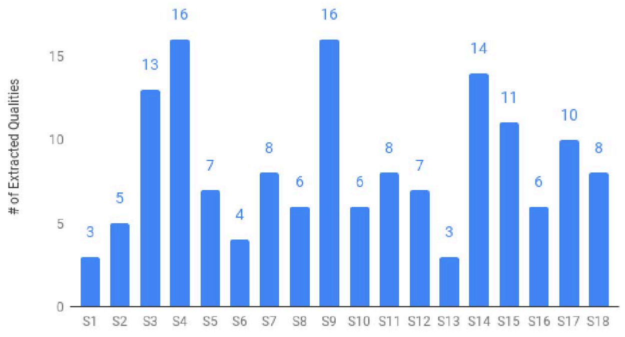


Fig. 2: Distribution of the Number of Qualities

were either extracted from standards, explained in previous works, or derived from concrete examples of how other sub-characteristics contribute to the particular quality attribute. Only the following 2 selected studies failed to provide any definitions on the mentioned quality attributes.

In S6, the authors identified challenges within the trade-off of various qualities and demonstrated the possibility of achieving a balance of trade-offs between high computational efficiency and other software quality attributes such as usability. However, the definitions for such quality attributes were not explicitly defined. In S14, the authors defined sustainability in two aspects. However, the quality attributes that contribute to the different dimensions of sustainability were not explicitly defined.

2) Consistency Among Definitions: The selected studies tended to use standards for defining the software qualities. The most common standard found was ISO/IEC 25010. While ISO/IEC 25010 is part of a larger standard, the SQuaRE series, it is differentiated in this mapping study since many of the studies specified that particular portion of the standard. Others included additional parts of the series such as ISO/IEC 25022 and ISO/IEC 25023. Figure 3 illustrates the distribution of the number of selected studies per each type of definition.

Accessibility was defined differently across certain studies. In ISO/IEC 25010, accessibility is defined as the "degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use" [18]. However,

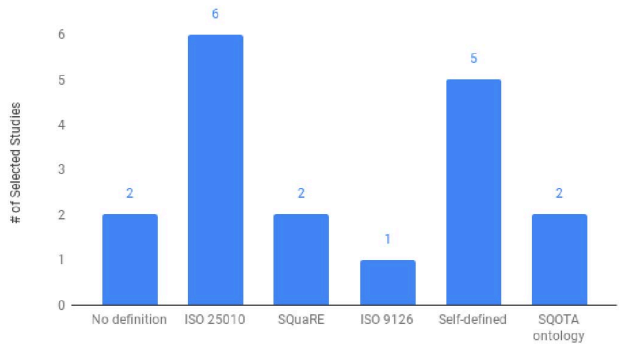


Fig. 3: Distribution of the Number of Selected Studies per Definition Basis

TABLE V: Efficiency

	Study ID
Efficiency	S3, S8, S10, S15, S16
Energy efficiency	S18
Resource efficiency	S18
Computational efficiency	S6
Life cycle efficiency	S9
Performance efficiency	S3, S11, S18

[21] [22] defined accessibility in the context of being available and reachable, which involves whether the intended areas of a software system can be accessed as desired. Similarly, [35] defined accessibility in the context of provenance, treating provenance as a resource, that can be accessed by the user.

In ISO/IEC 9126, efficiency was defined as "A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions" [38]. Later, efficiency is renamed "performance efficiency" in ISO/IEC 25010. We found that when people addressed efficiency in the selected studies, they tended to mention the specific kinds of efficiency. Table V shows the types of efficiency extracted from the selected studies.

Summary of RQ1. We extracted 71 distinct quality attributes from the selected studies. There has been a shift in the top qualities being considered within the context of quality interrelationships. The percentage of studies which do not provide definitions for the quality attributes has decreased within our selected set. While people are utilizing standardized definitions for software qualities, it suggested in S9 to consider the diversity of definitions across domains as the major weaknesses of the current software quality practices. There is also a rise in studies investigating one specific quality to provide a deeper understanding on the key activities and contributors associated with it. These qualities included maintainability [1] [22], sustainability [36] [37] [26], reliability [29], and provenance [35]. By doing so, these more complex qualities can become less ambiguous, so considering the various aspects individually may be a more productive approach.

B. RQ2

1) Study Purpose: Shown in Table III, of the selected studies, 12 reported on the actual quality interrelationships that were found through case studies, surveys, interviews, and literature (including standards, frameworks, and ontology).

S14 presents a multiple case study which looks to apply a sustainability model from their prior work. This model related software sustainability to quality requirements based on ISO/IEC 25010 using a web-based survey of researchers and practitioners. In their study, software sustainability is decomposed into technical, social, economic, and environmental dimensions; they identified quality attributes with dependency relationships with the different dimensions. Some quality attributes had dependencies on multiple dimensions such as modifiability which contributes to the economic, technical, and environmental dimensions.

In contrast, the remaining 6 studies reported on methods or frameworks for identifying or managing quality interrelationships. Both S4 and S16 proposed some frameworks to assist quality interrelationship identification. In S4, authors developed a process to support identifications for quality interactions. In S16, authors proposed an ontological approach to predict trade-offs between security and usability.

2) Evaluation: Shown in Table III, 7 papers did not present an evaluation of either the proposed method or the quality interrelationships being reported. The majority of the evaluations were based on case studies, including usage scenarios and applications of methods to example systems. Notably, S3 not only evaluated their approach for managing quality using an empirical study on 21 products, but they also performed a correlation analysis to evaluate the strength of the quality interrelationships. In S17, authors performed a large empirical study on identifying maintainability concerns from issue summaries and using the dependency relationship between issue summaries to identify the interrelationships among the quality attributes.

Summary of RQ2. There does not appear to be a strong preference for methods or relationship reporting studies. Case studies appear to be the preferred method of evaluation; however, overall, evaluation remains an issue for studies in this area.

C. RQ3

1) Software Development Stage: Shown in Table III, of the selected studies, 8 were applied at the earlier development stages, while 10 were from other stages. For example, the approach in S10 addressed software quality interrelationships for software components. Such components would be evaluated post development. Another example was found in S1 and S17, where the authors utilized bug reports, which are mostly generated during the maintenance phase, to illustrate the potential software quality interrelationships displayed through the "Block"/"Depends on" relationships among bug reports.

2) Automated Approaches: Of the selected studies, none extensively used tool support to identify quality interrelationships. The selected studies largely depended on expert ratings

TABLE VI: Classification of Interrelationships

Relationship Types	Study ID
Conflict	S2, S3, S4, S6, S7, S8, S16, S18
Synergy	S3, S8, S18
Means-End	S3, S5, S9, S10, S11, S12, S13, S14, S15, S18
Unknown	S1, S17

and judgement as input or to verify quality interrelationships. For example, studies S2 and S5 require experts to evaluate and perform pair-wise comparisons of qualities. Surveys, interviews, and questionnaires are also commonly used for eliciting expert and user opinions on qualities [26] [28] [32] [36] [37]. S10 reported on sub-characteristics based on various models and quality experts using Delphi method. Other studies developed or referenced ontologies and literature in their approaches [1] [24] [25] [33] [34] [35]. S1 used a classifier to identify software qualities expressed in issue summaries; the study relies upon "Block"/"Depends on" relationships being identified and reported with the issues. S3 mentions use of static analysis tools to measure quality sub-characteristics of maintainability, but these did not identify the relationships.

Summary of RQ3. These results could indicate a shift in focus from addressing software quality interrelationships at earlier stages of development to approaches that can be used at differing stages, with roughly equivalent numbers of studies being applicable to different stages of development. While it is considered beneficial to address conflicts earlier in the development life-cycle, two areas which may benefit from this change are in the handling of new quality requirements introduced during the development, operation, and maintenance stages as well as for dealing with conflicts in iterative development processes such as Agile. Automated, tool-assisted approaches were largely absent from our selected studies, and there remains a reliance on experts for addressing quality interrelationships.

D. RQ4

1) Types of Relationships: We identified the interrelationships between software qualities described in each study and classified the following types of interrelationships.

- Conflicts: increasing one quality decreases another
- Synergies: increasing one quality increases another
- Means-ends: breaking down higher level qualities into sub-quality attributes
- Unknown: the interrelationship could be any of the above

As shown in Table VI, means-end interrelationships have the most number of occurrences, followed by conflict interrelationships. Less than half of the studies that presented conflicts also reported synergy type interrelationships. None of the selected studies reported solely the synergy type.

S1 and S17 proposed an approach that determines quality interrelationships based on dependencies between bug reports. The qualities are related to each other by the bug report dependencies, so the interrelationship described might fit into any of conflict, synergy, or means-end. Thus we separated the studies into an "unknown" category.

We have identified 35 distinct synergy interrelationships and 54 conflict interrelationships. The most frequently mentioned synergy interrelationship is security and reliability (S2, S15 and S18). The most frequently mentioned conflict interrelationship is functional suitability and usability (S3 and S18).

Interestingly, we found that 4 interrelationships appeared in both synergy and conflict pairs. In S15, security and usability, usability and reliability are listed as synergies; however they appear as conflicts in S2. Similarly maintainability and reliability, portability and usability are listed as conflicts in S15; however they appear as synergies in S2 and S3 respectively.

2) Interrelationship Strength: Shown in Table III, of the selected studies, 7 considered some measurements of the strength of the interrelationships, which varied across studies. Studies S2 and S4 considered interrelationship strength based on relative impact: whether there is a positive, negative, or no impact between qualities. S5 used relative importance of qualities as defined by practitioners as part of their approach. Interrelationship strength was determined based on the Spearman's Rank correlation in S3 and S15. S18 additionally built regression models relating software qualities to different aspects of environmental sustainability.

Summary of RQ4. While the studies did not tend to use the language of examining synergies in their approaches, while attempting to address and evaluate conflicts, they identified synergies nonetheless. Quality attributes may relate differently in various contexts; some quality pairs identified in S15, which focuses on the sustainability domain, were opposite of S2 and S3. However, we did not see this occur for S6 or S11 which deal with the domains of high performance computing and Internet of Things respectively. Further work should be done to compare how much of an effect domain has on the types of quality interrelationships. In addition, almost half of the studies considered some type of measurement for the degree of inter-relatedness; however, there did not appear to be a predominant approach for measurement.

V. Conclusion

In this paper, we have reported a systematic mapping study of 18 carefully selected studies after 2015 to explore the new trends on the topic of software quality interrelationships. First, we have identified a set of research gaps gathered from previous reviews. We then proposed a number of research questions related to how software qualities are defined, the study purposes and evaluation, types of approaches, and the interrelationships these selected studies mentioned. We have sought to answer these questions and provided a systematic understanding of the selected studies, which have been discussed in depth against the research gaps. The overall conclusion that stems from our review results is that there have been improvements in the area of software quality interrelationships, yet a number of gaps still need to be filled.

In conclusion, we believe that our review is timely and important, as it reports on the results of recent trends in software quality interrelationships. More significantly, by comparing our findings to a set of research gaps identified from reviews prior to 2015, we gained understanding on what has been done and what still needs to be addressed.

References

[1] B. Boehm, C. Chen, K. Srisopha, and L. Shi, "The key roles of maintainability in an ontology for system qualities," in *INCOSE International Symposium*, vol. 26, no. 1. Wiley Online Library, 2016, pp. 2026–2040.

[2] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 311–317.

[3] D. Samadhiya, S.-H. Wang, and D. Chen, "Quality models: Role and value in software engineering," in *2010 2nd International Conference on Software Technology and Engineering*, vol. 1. IEEE, 2010, pp. V1–320.

[4] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Boston, MA: Springer US, 2000.

[5] K. B. Ijaz, I. Inayat, and F. A. Bukhsh, "Non-functional requirements prioritization: A systematic literature review," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2019, pp. 379–386.

[6] G. A. García-Mireles, M. Á. M. De La Rubia, F. García, and M. Piattini, "Methods for supporting management of interactions between quality characteristics," in *2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. IEEE, 2014, pp. 1–8.

[7] D. Mairiza, D. Zowghi, and V. Gervasi, "Conflict characterization and analysis of non functional requirements: An experimental approach," in *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)*. IEEE, 2013, pp. 83–91.

[8] D. Mairiza, D. Zowghi, and N. Nurmuliani, "Managing conflicts among non-functional requirements," in *Australian Workshop on Requirements Engineering*. University of Technology, Sydney, 2009, pp. 11–19.

[9] S. Barney, K. Petersen, M. Svahnberg, A. Aurum, and H. Barney, "Software quality trade-offs: A systematic map," *Information and software technology*, vol. 54, no. 7, pp. 651–662, 2012.

[10] M. Aldekhail, A. Chikh, and D. Ziani, "Software requirements conflict identification: review and recommendations," *Int J Adv Comput Sci Appl (IJACSA)*, vol. 7, no. 10, p. 326, 2016.

[11] R. B. Svensson, M. Host, and B. Regnell, "Managing quality requirements: A systematic review," in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2010, pp. 261–268.

[12] H. In and B. W. Boehm, "Using winwin quality requirements management tools: a case study," *Annals of Software Engineering*, vol. 11, no. 1, pp. 141–174, 2001.

[13] D. Mairiza, D. Zowghi, and N. Nurmuliani, "Towards a catalogue of conflicts among non-functional requirements." *ENASE*, vol. 2010, pp. 20–29, 2010.

[14] G. Me, C. Calero, and P. Lago, "A long way to quality-driven pattern-based architecting," in *European Conference on Software Architecture*. Springer, 2016, pp. 39–54.

[15] G. A. García-Mireles, M. Á. Moraga, F. García, C. Calero, and M. Piattini, "Interactions between environmental sustainability goals and software product quality: A mapping study," *Information and Software Technology*, vol. 95, pp. 108–129, 2018.

[16] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[17] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[18] "Iso/iec/ieee international standard - systems and software engineering–vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, Aug 2017.

[19] P. Bourque, R. E. Fairley, and I. C. Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014.

[20] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[21] C. Chen, M. Shoga, and B. Boehm, "Exploring the dependency relationships between software qualities," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, July 2019, pp. 105–108.

[22] C. Chen, S. Lin, M. Shoga, Q. Wang, and B. Boehm, "How do defects hurt qualities? an empirical study on characterizing a software maintainability ontology in open source software," in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2018, pp. 226–237.

[23] N. Tsuda, H. Washizaki, K. Honda, H. Nakai, Y. Fukazawa, M. Azuma, T. Komiyama, T. Nakano, H. Suzuki, S. Morita *et al.*, "Wsqf: Comprehensive software quality evaluation framework and benchmark based on square," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 312–321.

[24] B. Boehm, "Improving and balancing software qualities," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 890–891.

[25] W. Roh and S.-W. Lee, "An ontological approach to predict trade-offs between security and usability for mobile application requirements engineering," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2017, pp. 69–75.

[26] S. A. Koçak, G. I. Alptekin, and A. B. Bener, "Integrating environmental sustainability in software product quality." in *RE4SuSy@ RE*, 2015, pp. 17–24.

[27] M. A. Al Imran, S. P. Lee, and M. M. Ahsan, "Measuring impact factors to achieve conflict-free set of quality attributes," in *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*. IEEE, 2017, pp. 174–178.

[28] G. A. García-Mireles, M. Á. Moraga, F. García, and M. Piattini, "A process support with which to identify interactions between quality characteristics," in *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 2015, pp. 21–39.

[29] F. Febrero, C. Calero, and M. Á. Moraga, "Software reliability modeling based on iso/iec square," *Information and Software Technology*, vol. 70, pp. 18–29, 2016.

[30] D. Pflüger and D. Pfander, "Computational efficiency vs. maintainability and portability. experiences with the sparse grid code sg++," in *2016 Fourth International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering (SE-HPCCSE)*. IEEE, 2016, pp. 17–25.

[31] T. Bi, P. Liang, and A. Tang, "Architecture patterns, quality attributes, and design contexts: How developers design with them," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018, pp. 49–58.

[32] M. Wahler, R. Eidenbenz, A. Monot, M. Oriol, and T. Sivanthi, "Quality attribute trade-offs in industrial software systems," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 2017, pp. 251–254.

[33] A. Tiwari and P. S. Chakraborty, "Software component quality characteristics model for component based software engineering," in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*. IEEE, 2015, pp. 47–51.

[34] J. J. Tambotoh, S. M. Isa, F. L. Gaol, B. Soewito, and H. L. H. S. Warnars, "Software quality model for internet of things governance," in *2016 International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 2016, pp. 1–6.

[35] A. L. de Castro Leal, J. L. Braga, and S. M. S. da Cruz, "Cataloguing provenance-awareness with patterns," in *2015 IEEE Fifth International Workshop on Requirements Patterns (RePa)*. IEEE, 2015, pp. 9–16.

[36] N. Condori-Fernandez and P. Lago, "Towards a software sustainability-quality model: Insights from a multi-case study," in *2019 13th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2019, pp. 1–11.

[37] S. Aljarallah and R. Lock, "An exploratory study of software sustainability dimensions and characteristics: end user perspectives in the kingdom of saudi arabia (ksa)," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[38] "Software engineering – product quality," International Organization for Standardization, Standard ISO/IEC 9126-1:2001, 2001.