Kuwait J.Sci., Vol.49, No.(1),January.2022,pp(1-43)

Assessing the impact of socio-technical congruence in software development: a systematic literature review

Binish Raza[1], Rodina Ahmad[1,*], Mohd H.N.M Nasir[1], Shukor S.M Fauzi[2], Muhammad A. Raza[3]

[1]*Faculty of Computer Science and Information Technology, Universiti Malaya (UM), 50603 Kuala Lumpur, Malaysia*

[2]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perlis, Malaysia*

[3]*Dept. of Information Technology, Bahauddin Zakariya University, Multan, Pakistan*

[*] *Corresponding Author: rodina@um.edu.my*

Abstract

Software development is a critical task that depends on coordination among team members and organizational activities that bring team members together. The literature indicates various techniques that have been applied to control the coordination level among team members. Notable among these techniques is social-technical congruence (STC), which helps to measure the alignment between the social and technical capabilities of an organization and teams at various stages of software development. The dynamic nature and changes of coordination requirements make STC a potential research area in this regard. The main objective of this study is to perform a systematic literature review (SLR) that recognizes and structures existing studies that represent new evolutionary trends in the field of STC. A SLR is performed of 46 publications from 4 data sources, including journals, conferences and workshop proceedings, most of which were published between 2008 and 2019. To this end, a thorough analysis is carried out to elicit the studies based on 7 research questions in this SLR. The outcome of this SLR is a set of ample research studies representing various aspects, performance impacts, factors, and evolutionary trends in the field of STC. Furthermore, STC measurement techniques are classified in two distinct groups, matrix based and social network analysis-based measures. After a systematic exploration of these aspects, this study results in new insightful features and state of art of STC. This SLR concludes that some areas still require further investigation. For instance, (1) STC-related literature exists, but only one research work mainly focuses on the risk of overwhelming STC (i.e., excessive STC measurement may overburden the software development process); (2) STC measurement techniques facilitate the identification of congruence gaps, but no attention has been given towards the unweighted social network analysis based STC measurement models; (3) STC measurement techniques are generally applied in the development phase of the project lifecycle, but these measurements are rarely used in other software development phases, such as the requirement and testing phases or all phases; and (4) The development factors that effects on STC measurement are rarely focused by researchers in the context of various domains.

Keywords: Social dependency; socio-technical congruence; socio-technical coordination; sociotechnical dependency; software development; technical dependency

1. Introduction

The complex and time-consuming nature of today's software development tasks increases the time to project completion. This aspect raises the overall cost of projects and the need to improve developers' skills (Hameed *et al.*, 2017; Tahir *et al.*, 2016). To deal with multifaceted and time-intensive tasks, one common solution is to distribute the tasks between software project team members who may be working from geographically separated locations. The team distribution and modular nature of software development reduce labor costs but also increases the risks related to team collaboration and coordination (Dingsøyr *et al.*, 2017; Alqarni & Qureshi, 2019). Despite over 30 years of research and evolution of software development tools and techniques, problems and failures are common in the domain of distributed software development (DSD) (Abbasi *et al.*, 2019; Alizzaf, 2015). Thereby, DSD is still an intense research area.

In a distributed environment, the main tasks are to understand the flow of information among teams (Datta, 2018; Yahyaoui *et al.*, 2020) and identify key individuals in the communication network (Amirfallah *et al.*, 2019). Teamwork facilitates quick software development while also adding overwhelming load in terms of task interdependencies with complex technical mechanisms of software development (Ibrahim *et al.*, 2018). The issue of task interdependency can be addressed by introducing social capabilities into the software development process (i.e., communication among team members). Communication serves as a connector between developers who need coordination to complete interdependent tasks and achieve the same goal (Kuhrmann & Münch, 2016). The organization of communication and coordination activities is a critical task that not only impacts team performance but also influences product quality.

One well-known technique of overcoming problems with appropriate communication and coordination is enhancing socio-technical congruence (STC). This technique involves measuring and aligning the social and technical capabilities of an organization to support the multiple levels of software development. In the literature, STC is defined as a fit between task dependencies and the coordination activities among the people in an organization (Sobri *et al.*, 2017). STC focuses on the technical and social aspects of the software development process, and fit indicates the right fusion of technical and social abilities within a distributed team.

STC helps measure the level of team coordination, which subsequently assists an organization in identifying gaps that cause delays in work and result in overall project failure (Zhang *et al.*, 2018). Many existing studies address the impact of STC on the success of software development. Above all, a systematic literature review (SLR) (Snyder, 2019) is the best choice to produce a synthesized view of these existing studies, which would assist researchers in capturing new interpretations of STC. Although STC has received attention in recent research efforts on software development, no work has systematically reviewed such studies, especially in terms of STC impact on software quality and team performance. To the best of our knowledge, one previous study conducted a systematic literature review in the field of STC. One of the objectives of this SLR is to investigate the impact of the congruence between technical and social capabilities in software development.

The current SLR synthesizes various aspects of STC, distinguishes various factors that influence STC calculation, identifies the data sources used for STC measurement, determines the impacts of STC on software development outcomes, and presents the effects of the nonexistence of STC. Furthermore, this work covers the latest STC trends. In this SLR, the literature up to the year 2019 was gathered from four popular digital databases: Web of Science (WoS), IEEE, ACM, and Scopus. The studies selected from literature are validated using quality assessment criteria, which is designed with the consensus of the authors and various software engineering experts. This SLR also considers the empirical, theoretical, and case study papers, as well as experimental surveys on STC.

The main contributions of this work are as follows:

- STC concept addressed in the literature is synthesized and existing techniques to measure STC are analyzed and presented. The SLR also highlights the components, data sources, and challenges of STC measurement, thereby assisting organizations and software developers to become more aware of congruence issues in the hope of reducing risks throughout the software development process.
- Quality assessment scores are stipulated for the selected STC studies to ensure the inclusion of high-quality studies from the literature.
- Prospective directions in the field of STC that lack attention from the research community are identified.

This paper is divided into several sections. The first section outlines the introduction that highlights the background, issues, and motivation of this SLR. Section 2 discusses the SLR methodology used to achieve the proposed objective. Next, the results and a detailed discussion are presented in sections 3 and 4 respectively. Section 5 discusses some threats to the validity of this work. The final section concludes this study with some future directions.

2. Research methodology

A SLR is a way to discover, assess, and interpret existing studies interrelated to research questions, fields, or trends. The main objective of an SLR is to gather evidence from related studies to deduce results.

In Kitchenham's work (Kitchenham & Charters, 2007; Kitchenham, 2004), the systematic review methodology is described as a collection of various activities that help to accomplish three core stages: planning, conducting and reporting the SLR. To conduct this SLR, the guidelines provided by Kitchenham are followed. Furthermore, the software tool StArt (State of the Art through Systematic Reviews) (Fabbri *et al.*, 2016) is used to support and validate the current research methodology. The complete review process consists of several steps as described in the following subsection.

2.1 Review procedure

The entire review process consists of three stages which further contain several activities (as shown in Figure 1). The main objective of this review is to understand the research background and identify new trends in STC to facilitate further investigation.

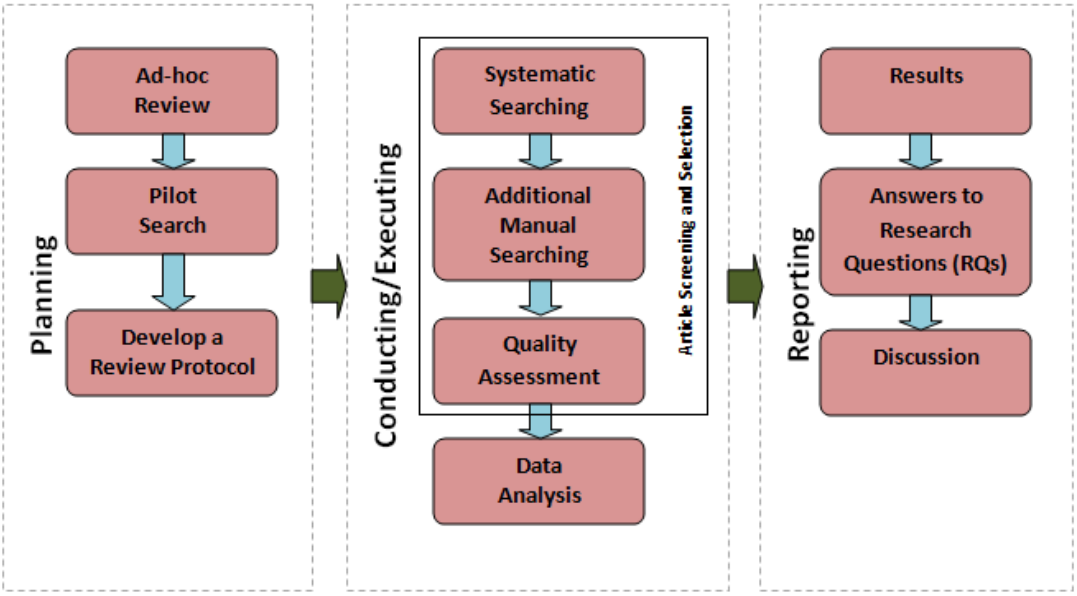

Fig. 1. Review procedure

2.1.1 Planning

The planning stage consists of three main activities that help to systematically plan the overall approach to conduct this SLR.

1) Ad-hoc literature review

An ad-hoc literature review on STC is performed to recognize its impact on software quality and distributed team performance. To this end, an investigation of relevant articles is conducted first. The search process focused on the articles related to STC and its impact on software development. Ad-hoc review papers serve as a guideline to develop a review protocol and formulate the research questions in this SLR. However, some studies obtained as an output of this pilot search (discussed in the following section) are included in this SLR along with additional papers obtained after the screening and selection processes (discussed in subsection (f) of review protocol).

2) Pilot search

To efficiently perform this systematic review, a review protocol was developed. The proposed protocol helped to identify search terms, data sources, criteria for studies inclusion and exclusion, quality assessment criteria, and the designing of data extraction form. Further, the review process was refined by performing a pilot search on a specified selection pool consisting of articles published over a decade from 2008 to 2019. From the ad-hoc review, it was discovered that the

most significant and highest number of publications related to STC was from the year 2008 (Sierra *et al.*, 2018). For this reason, the selected search period started in 2008.

3) Review protocol

The proposed review protocol comprises six steps (Figure 2) to help perform the SLR systematically. The detail of each step is given in the following subsections.

a) Research questions

The purpose of this SLR is to identify new trends, techniques, and factors to measure STC in software development that would improve team performance and software quality.

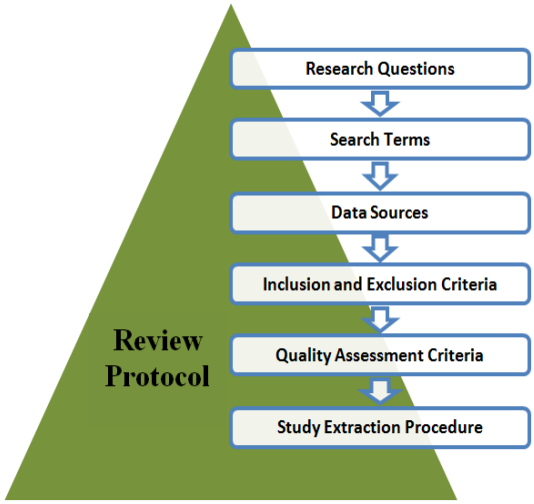

Fig. 2. Review protocol

The following research questions (RQs) were derived from the objective of this SLR.

RQ1.What are the major components of STC?
RQ2. How did STC evolve over the period from 2008 to 2019?
RQ3.What is the impact of STC on software development outcomes?
RQ4.What are the consequences of non-congruence?
RQ5.What socio-technical data sources are used in literature for STC measurement?
RQ6. What techniques exist to measure STC?
RQ7.Which factors influence STC measurement?

b) Search terms

The ad-hoc search helped identify relevant search terms to extract significant studies. From the SLR RQs, a list of search terms was identified (Table 1). An initial search with the identified terms presented numerous irrelevant studies that are not pertinent to software engineering but rather to social science, artificial intelligence, etc. To refine the search to exclude irrelevant studies, contextual terms were added to the search term list, such as software development. The search terms were adjusted to cover as much as possible and also to validate the search accuracy (Kitchenham & Charters, 2007). Several combinations were first tried in searching for relevant

studies, and the outcome was compared with the initial search pool. Finally, the search query was formulated using the Boolean "AND" and "OR" operators with the key terms selected. The resulting search query was as follows:

("Socio-technical congruence" OR "Socio technical congruence" OR "Sociotechnical congruence" OR "Social-technical dependenc "OR "Social technical dependenc*" OR "Socialtechnical dependenc*") OR ("Social dependenc*" AND "Technical dependenc*") OR (("Social technical Coordination") OR (("Software development") AND ("Team Coordination" OR "Team Collaboration" OR "Team Communication")))*

The search query was adapted according to the searched data source limitations (i.e., where a complex or long search query was not supported). The applied adaptation is found in Appendix A.

Table 1. Search terms

Criteria	Keywords
Socio-Technical Congruence	Socio-technical congruence
	Socio technical congruence
	Sociotechnical congruence
	Social dependenc*
	Technical dependenc*
	Social-technical dependenc*
	Social technical dependenc*
	Social-technical Coordination
	Social technical Coordination
Software Development	Software Development
	Team Coordination
	Team Collaboration
	Team Communication

c) Data sources

The Web of Science (WoS) core collection, Scopus, IEEE, and ACM were considered as the primary data sources of relevant studies on STC because they include the key publications regarding the targeted research. The databases with search criteria for collecting the results are presented in Table 2. Overall, 449 papers were gathered between 2008 and 2019 from the data sources using the selected keywords.

Table 2. Data sources with search criteria

Digital Source	Criteria
IEEE Explore	2008-2019, Journals, Conferences, Early Access Articles
Web of Science	2008-2019, Proceedings Papers, Articles, Reviews
Scopus	2008-2019, Conference Papers, Articles, Book Chapters
ACM	2008-2019, Conference Papers, Articles

The search procedure focused on keywords based on the objective of this SLR to help identify studies related to STC and its new trends. In the first stage, 59 duplicate studies were excluded

from the gathered results, and 390 studies were left in the selection pool. Subsequently, a systematic screening of the remaining papers was conducted by reading the titles, keywords, and abstracts; the output was 181 papers. The results of the initial and selected study pools are shown in Figure 3. The full text screening yielded 60 selected studies. To accommodate for the gray literature, the reference lists of the selected studies were also investigated, thus adding 18 more papers to the selected pool and making a collection of 78 articles. After qualifying the inclusion, exclusion, and quality assessment criteria, a total of 46 papers were collected according to the study objective.

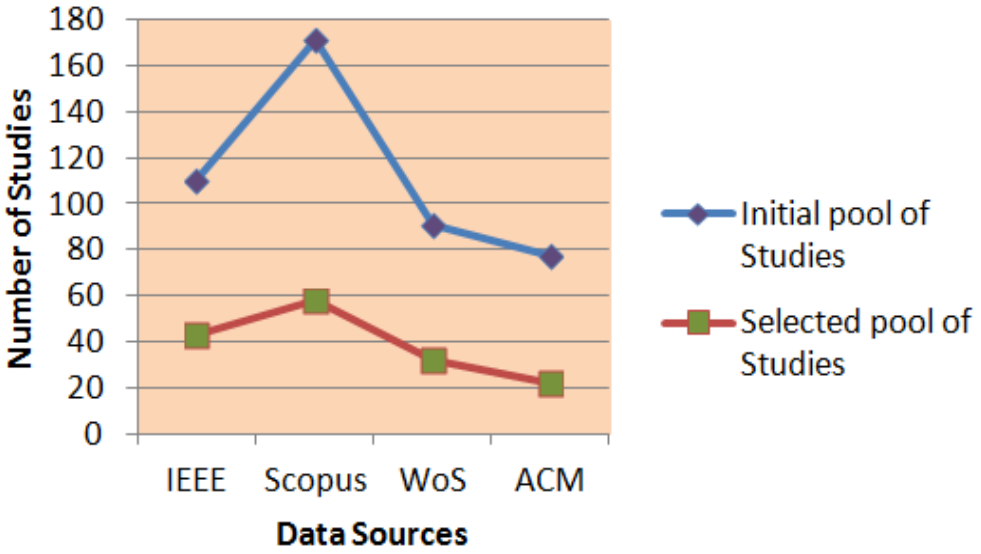

Fig. 3. Proportion of initial and selected study pools

d) Inclusion and exclusion criteria

The main goal of inclusion and exclusion criteria is to investigate the studies that are directly related to the research questions and to lessen the probability of bias (Kitchenham & Charters, 2007). The inclusion and exclusion criteria applied to the present SLR are listed in Tables 3 and 4. These criteria were applied to all studies during the study extraction procedure (see the following section f).

Table 3. Inclusion criteria

Identifier	Criteria
I1	Papers directly related to STC and written in English
I2	Published papers at the journal, workshops, and conferences that directly relate to the research objective
I3	Studies that present some contribution on the use of STC measurement to support software development activities and also that satisfy the minimum quality threshold
I4	Inclusion of early key cited articles
I5	Articles published from 2008 to 2019

Table 4. Exclusion Criteria

Identifier	Criteria
E1	Studies wrote other than in English
E2	Studies other than proceedings, reviews and journal papers
E3	Papers that do not focus on STC
E4	Documents that claim to use STC but whose aim is not to illustrate STC itself nor to explain STC techniques or to identify tools for measuring STC.
E5	Articles where the search keywords are listed only in the references

e) Quality assessment criteria

Quality assessment criteria (QAC) ensure accurate and relevant data extraction from the primary selected studies. For this review, five criteria were formulated (comprising nine quality test questions) according to the research scope and to evaluate the relevance of existing studies to this SLR. Details of these QAC are given in Table 5. The quality assessment for each final study chosen was attained by computing a score on the basis of objective reporting (problem statement), approach clarity (research design), sample data and tool selection (data collection), approach validation (data analysis) and result reliability (conclusion).

Table 5. Quality assessment criteria

Identifier	Quality Criteria
Problem Statement	
Q1	Are the goals and objectives clearly specified?
Q2	Is there a sufficient explanation of the basic studies?
Design	
Q3	Does the research design support the study objectives?
Q4	Is the proposed technique clearly described?
Data Collection	
Q5	Is the study supported by a tool?
Q6	Are the measures used in the study applicable for answering the research question?
Data Analysis	
Q7	Is there any validation of the proposed method/approach?
Conclusion	
Q8	Are the study findings clearly affirmed and supported by the results?
Q9	Is there a clear statement of the limitations and contributions?

The QAC score of each test question was determined using a three-grade scale (full=1, partial=0.5, and none=0) as depicted in Table 6. The quality test is assessed by the first author and then verified by the other authors. It is verified that the selected studies discussed the issues related to each quality criterion or not. Any differences among the authors' opinion are resolved by discussion until a common consent was entrenched. The question was scored as: If the study completely answers the question, the score of associated QA is 1, a partial answer scores 0.5, and 0 means the test question is not addressed in the study.

Table 6. Testing studies on quality assessment criteria

Paper ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	QAC Score
PS1	1	0.5	0.5	0	0.5	0.5	0.5	0.5	1	5.5
PS2	1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	5.5
PS3	1	0.5	1	0.5	0.5	1	0.5	0.5	1	6.5
PS4	1	0.5	1	1	0	0.5	0.5	0.5	0.5	5.5
PS5	1	1	1	0.5	0	0.5	0	1	1	6
PS6	1	0.5	0.5	1	0	1	0	0.5	1	5.5
PS7	1	0.5	0.5	1	1	0.5	0.5	1	0.5	6.5
PS8	1	1	1	1	0.5	0.5	0.5	0.5	1	7
PS9	1	1	1	1	0.5	1	0.5	0.5	1	7.5
PS10	1	1	1	1	0.5	1	0.5	1	0.5	7.5
PS11	1	1	1	1	0	0.5	0	0.5	1	6
PS12	1	0.5	1	1	0.5	1	0.5	1	0.5	7
PS13	1	1	0.5	0.5	0	0.5	1	1	0.5	6
PS14	1	1	0.5	0.5	1	0.5	0.5	0.5	1	6.5
PS15	1	1	1	1	0	1	0	1	0.5	6.5
PS16	0.5	1	1	1	0	1	0.5	0.5	1	6.5
PS17	1	1	0.5	1	0.5	0.5	0.5	0.5	0	5.5
PS18	0.5	1	1	1	0	1	0.5	0.5	1	6.5
PS19	1	0.5	1	1	0.5	0.5	0.5	1	0.5	6.5
PS20	1	0.5	1	1	0	1	0.5	1	1	7
PS21	1	0.5	0.5	0.5	0.5	1	0.5	1	0.5	6
PS22	1	0.5	1	0.5	0.5	0.5	0.5	1	0.5	6
PS23	1	1	1	1	0.5	1	0.5	1	0.5	7.5
PS24	1	1	0.5	0.5	0.5	1	0.5	0.5	0	5.5
PS25	1	0.5	1	1	0.5	1	0.5	1	1	7.5
PS26	1	1	1	1	0.5	1	0.5	0.5	1	7.5
PS27	1	1	0.5	1	1	1	0.5	1	1	8
PS28	1	1	1	1	0.5	1	0.5	1	1	8
PS29	1	0.5	1	1	0.5	0.5	0.5	0.5	1	6.5
PS30	1	0.5	1	1	0.5	1	0.5	1	1	7.5
PS31	1	0.5	1	1	1	0.5	0.5	1	0.5	7
PS32	1	0.5	1	1	0.5	0.5	0.5	0.5	0.5	6
PS33	1	0.5	1	1	0.5	0.5	0.5	1	0.5	6.5
PS34	1	0.5	0.5	1	0	0.5	0.5	1	0.5	5.5
PS35	1	0.5	0.5	1	0	0.5	0.5	0.5	0.5	5
PS36	1	0.5	1	1	0.5	0.5	0.5	1	1	7
PS37	1	0.5	1	1	0.5	0.5	0.5	0.5	0.5	6
PS38	1	1	1	1	0.5	0.5	0.5	1	0.5	7
PS39	1	0.5	1	1	0.5	0.5	0.5	1	0.5	6.5
PS40	1	1	1	1	0.5	0.5	0.5	1	0.5	7
PS41	1	1	1	1	0.5	0.5	0.5	1	0.5	7
PS42	1	1	1	1	1	0.5	0.5	1	0.5	7.5
PS43	1	1	1	1	0.5	1	0.5	0.5	1	7.5
PS44	1	0.5	0	1	0.5	1	0.5	1	1	6.5
PS45	1	1	1	1	0.5	1	0.5	0.5	1	7.5
PS46	1	1	0.5	1	0.5	1	0.5	1	0.5	7

Figure 4 indicates the cumulative QAC representation of each study against three scores.

To ensure the quality and reliability of the selected studies, only studies with above-average quality scores were considered (the threshold was set to 60%). Furthermore, the non-fulfillment of any two QAC was tolerable in selecting a particular article. The proposed mechanism strengthened the evidence reported in STC literature and ensured the selection of high-quality studies relevant to STC for investigation.

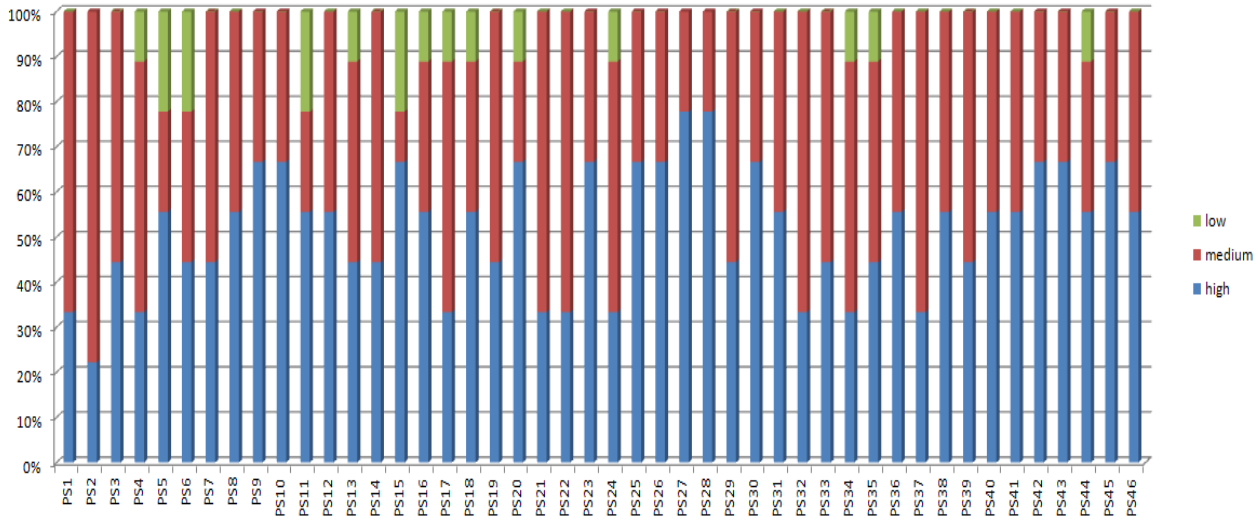

Fig. 4. QAC cumulative score graph for selective studies

Figure 5 shows the percentage of the proposed selected and rejected studies based on the QAC scores.

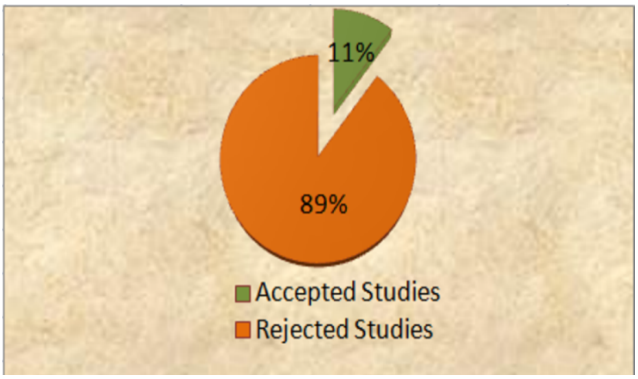

Fig. 5. Proposed accepted and rejected studies

f) Study extraction procedure

Data extraction was accomplished by a thorough collaboration of the main and co-authors. Following a detailed discussion and consensus of the team of authors, a study analysis procedure was devised for the screening (i.e., analysis of the introduction, full content, and conclusion sections) of each candidate study. To perform this procedure systematically, a data extraction form was designed iteratively (Appendix B) (Li *et al.*, 2015). The data fields were extracted from the

selected studies according to the designed form. First, a general form was constructed to extract data fields (e.g., publication source, year of publication, study type, domain scope, methodology, framework, software project team distribution, and project outcomes) from the selected studies. Then, the form fields were refined iteratively during pilot data extraction (i.e., the inclusion of research questions being addressed, team coordination method adopted, and team coordination measure fields). The extraction form refinement process continued until a saturation point was reached. The entire mechanism helped address the research questions via a multistage analysis of the full context of the articles under review.

2.1.2 Conducting/Executing

Once the review protocol has been developed and agreed by all authors, the proper execution of SLR started which further consists of several steps as discussed in the following subsections.

1) Article screening and selection

The screening phase in the current methodology comprises three key steps. First is the systematic search step to identify the initial STC-related studies. These initial articles were further screened to extract more relevant STC studies. The second step is known as the additional manual search, in which the cited references of relevant studies were searched manually. The third step, quality assessment, entails testing the eligibility of relevant studies using the defined criteria. Each step was performed according to a detailed consent meeting to guarantee further confidence and the least biases in the study inclusion process. The overall article screening and selection procedure are presented in Figure 6.

a) Systematic search

Following the review protocol, study identification was performed through a systematic search that produced a set of 449 published papers. Of these, 59 duplicate articles were excluded from the initial pool of studies selected. On the remaining pool, screening was performed using article titles, abstracts, and keywords, with an outcome of 181 papers left in the pool. Following a thorough analysis of the full paper texts, 60 papers were finally selected from the article pool for future investigation.

b) Additional manual search

To accommodate the gray literature (i.e., not available through the usual bibliographic sources, such as the databases searched for this review), a manual search (snowballing) was additionally performed on the cited reference list of 60 papers selected, yielding 18 additional articles.

c) Quality assessment

The quality assessment validates the selection of high-quality studies for a systematic review. In this step, two pools of articles were taken into account (i.e., one pool of 60 articles obtained as part of the systematic search and another pool of 18 papers extracted using snowballing). The articles

that did not efficiently fulfill the defined quality assessment criteria were excluded (as discussed in the quality fifth step of the review protocol). From the pool of 60 articles, only 34 articles fulfilled the quality assessment criteria, whereas 12 papers were filtered out from the pool of 18 papers according to the quality criteria defined, which resulted in 46 (34 + 12) primary studies in the final selection pool, which should be investigated further.

2) Data analysis

After the primary selection of STC studies, the relevant concepts were extracted from the selected pool of articles using the proposed study extraction procedure (subsection 1 under section 2.1.2). The review team performed a multistep analysis of each STC study selected to achieve the answers to the research questions in this SLR. The analysis outcomes are discussed in section 2.1.3.

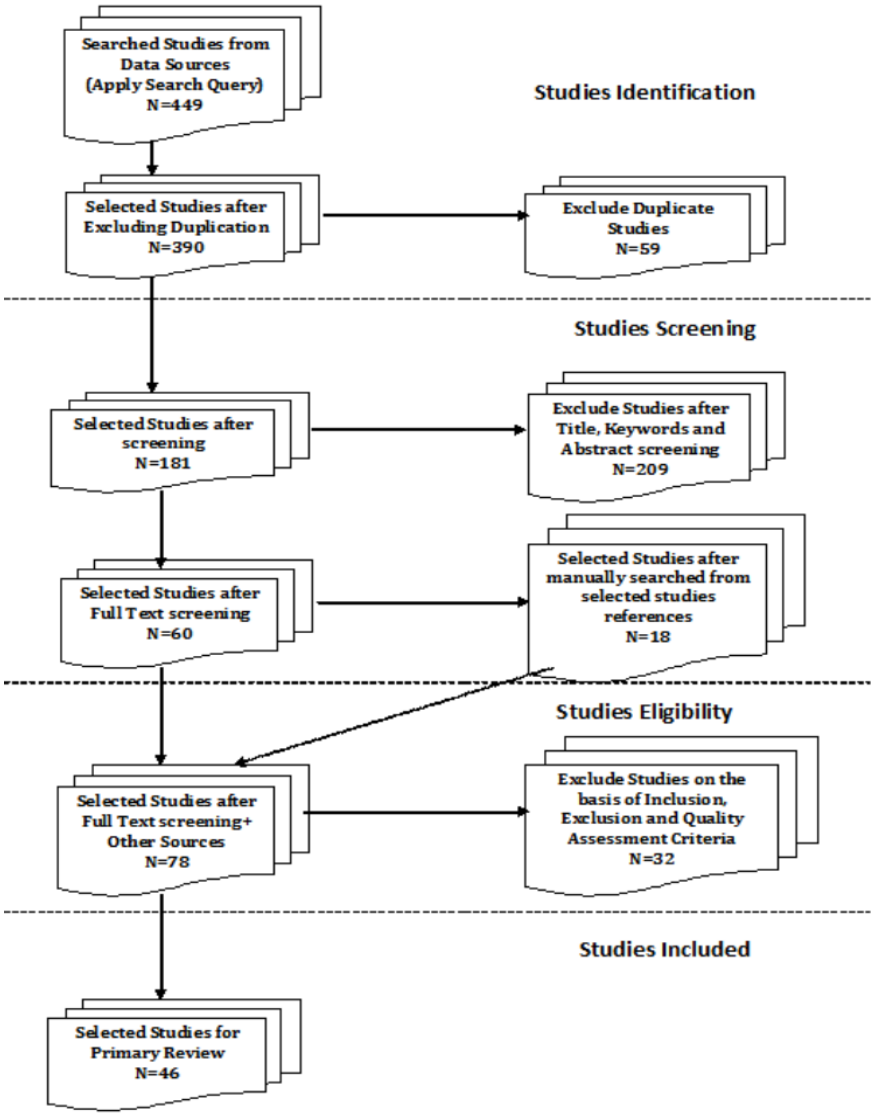

Fig. 6. Study Screening and Selection Steps

- *Demographic data and overview*

Before creating a report of the analyzed data, the STC data from the selected studies are synthesized and presented as a demographic overview. An STC demographic study is important because it provides an overview of researchers' contributions in the field of STC over time. Figure 7 demonstrates the total number of STC studies published by four data sources in the period between 2008 and 2019. It can be deduced that the peak publication years were from 2015 to 2018, and many STC-related works were published in Scopus and WoS. Among the four data sources, ACM offers the lowest number of STC studies over the 2008-to-2019 period.

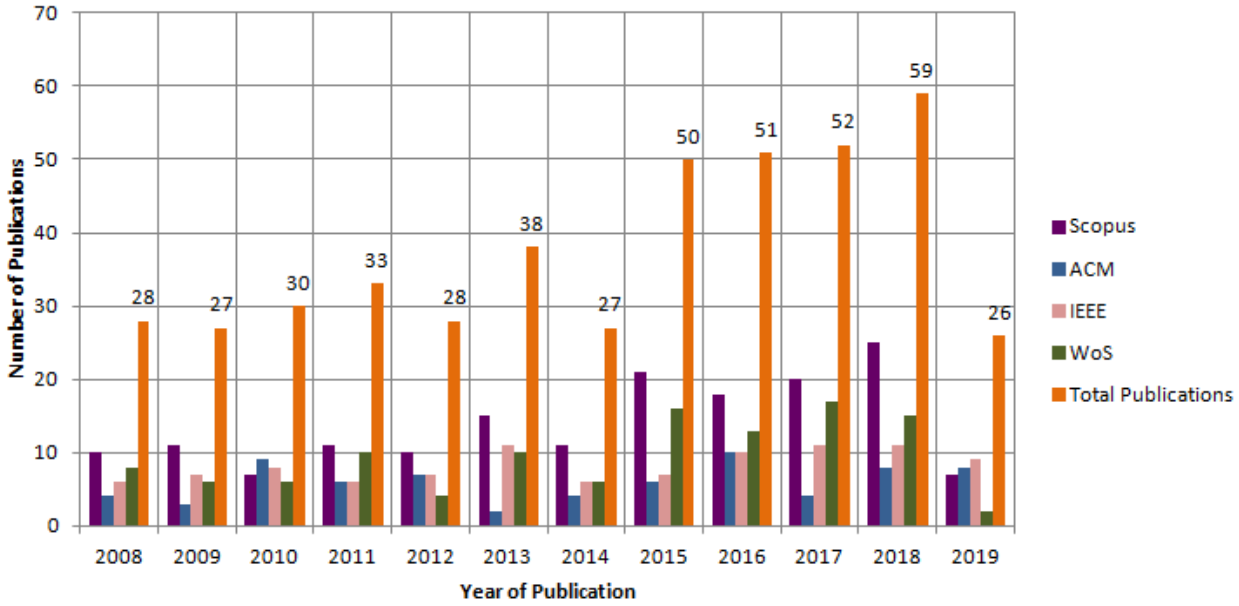

Fig. 7. Number of publications per year

Table 7 provides interesting demographic information on the 46 studies selected in terms of their focus on addressing the proposed research questions (RQs). It is observed that many studies answer the seven RQs. For instance, 15 studies (32%) deal with RQ1, 40 studies (87%) refer to answer RQ2, 16 studies (35%) report on RQ3, 11 studies (24%) answer RQ4, 26 studies (57%) address RQ5, 12 studies (26%) discuss RQ6, and 13 studies (28%) relate to RQ7.

2.1.3 Results

The report was generated after a thorough analysis of selected studies based on 7 proposed RQs. The detail of results is discussed in the following section

1) Overview

To investigate how the literature approaches the concept of STC, the related studies were synthesized via a review procedure with the defined review protocol. A study analysis was then

adopted to extract relevant concepts from the selected studies to address the RQs of this SLR. A complete report on the selected studies is found in Appendix C. This section presents a detailed analysis of the primary studies selected in the context of the proposed RQs.

2) Outstanding Aspects of STC

In this section, we have summarized the extracted information that represents the main findings of SLR by answering each RQ (as defined in our review protocol).

RQ1. What are the major components of STC?

The literature contains abundant evidence of STC. To increase STC coverage, interconnected studies from relevant references of selected publications were included. Each study provides its conceptualization of STC and highlights the general components. A generic congruence model is primarily thought to be the reflection of Conway's Law (Conway, 1968) which asserts that the organizational design is a replica of the underlying organizational communication structure.

Through a detailed analysis of literature, the authors in (Wyman, 2003) identified three major components of a congruence model: input, transformation process, and output.

Table 7. Demographic information and overview of the selected studies

Paper Id	Identifier	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7
PS1	M Maheshwari *et al.* (2012)	√		√		√		
PS2	S Datta (2018)	√	√	√		√		√
PS3	RS Sangwan(2008)	√				√		
PS4	M Cataldo *et al.* (2008)					√		
PS5	M Cataldo *et al.* (2006)	√		√		√	√	
PS6	M Cataldo *et al.* (2013)		√	√		√		√
PS7	J. Portillo Rodríguez *et al.*(2013)		√					√
PS8	M Cataldo *et al.* (2008)	√	√	√		√	√	√
PS9	A Sarma *et al.* (2008)	√	√		√			
PS10	LJ Colfer *et al.* (2016)		√	√				
PS11	G Valetto *et al.* (2007)	√		√		√	√	
PS12	X Wang(2018)		√					
PS13	I Kwan *et al.* (2011)	√	√	√		√	√	
PS14	J Portillo *et al.* (2014)		√	√		√	√	
PS15	Irwin Kwan *et al.* (2009)		√				√	
PS16	D Šmite *et al.* (2017)		√					
PS17	Kate Ehrlich *et al.* (2008)		√		√	√		
PS18	F Bolici *et al.* (2009)		√			√		
PS19	D Šmite *et al.* (2012)	√	√		√	√		
PS20	Li Jiang *et al.* (2012)	√	√			√	√	√
PS21	B Gokpinar wt al. (2010)		√		√		√	
PS22	G Valetto *et al.* (2008)		√	√				
PS23	T Dingsøyr *et al.* (2018)		√		√			

PS24	ML Bernardi *et al.* (2012)		√		√	√		
PS25	Stefanie Betz *et al.* (2013)		√		√			
PS26	ME Sosa *et al.* (2004)				√			
PS27	S Marczak *et al.* (2009)	√	√	√	√	√		
PS28	M Cataldo *et al.* (2009)		√		√			
PS29	W Sobri *et al.* (2017)		√	√		√		
PS30	I Kwan *et al.* (2011)	√	√			√	√	√
PS31	F Calefato *et al.* (2017)		√					√
PS32	M Cataldo *et al.* (2008)		√					√
PS33	C Bird *et al.* (2011)	√	√					√
PS34	L McLeod *et al.* (2011)		√					√
PS35	ZUR Kiani *et al.* (2013)		√					√
PS36	T Nguyen *et al.* (2009)		√					√
PS37	P Wagstrom *et al.* (2010)		√	√		√	√	
PS38	Mark S. Avnet (2016)		√				√	
PS39	N Bettenburg *et al.* (2011)		√	√		√		
PS40	AD de Santana *et al.* (2013)		√					
PS41	AJ Suali *et al.* (2017)		√	√		√		
PS42	MM Syeed *et al.* (2013)	√	√		√	√		
PS43	M Golzadeh (2019)		√			√		
PS44	M Palyart *et al.* (2018)		√			√		
PS45	W Zhang *et al.* (2019)	√	√	√		√	√	
PS46	D. A. A. Tamburri *et al.* (2019)		√			√		√
	Relative amount of publications	15	40	16	11	26	12	13
	Weighted of related publications	32%	87%	35%	24%	57%	26%	28%

However, the input comprises different elements including the organization environment, human resources, and history. Furthermore, the transformation process can consist of teamwork, people (who do this work), and the organizational structure and culture that transform organizational input to output in terms of quality and performance measurements. The fit is the alignment of relationships among all of these elements. Keeping in view the general congruence model, the first operational STC model was presented by Cataldo *et al.* (2008) who defined it as a fit between the socio (consisting of people and culture) and technical (including work and organizational structure) aspects of an organization. Concerning various development domains such as industrial, collaborative software development (CSD), DSD, GSD, and OSS, various studies highlight their socio and technical aspects as input for the STC model and provide the output in terms of organizational performance.

For instance, Maheshwari *et al.* (2012) discussed five major components of the STC model to facilitate information system management in the industrial domain. The model comprises (1) technical capabilities, (2) social capabilities, (3) team performance, (4) a development team, and (5) a fit (calculated from profile deviation) among all of these. Social capabilities focus on interpersonal interactions, regulations, and standards that characterize these interactions. However, technical capabilities concern two main aspects of software development: software design and planning. To measure the congruence between social and technical capabilities, the proposed study

focused on the profile deviation approach (indirect measurement) to evaluate the alignment among a development team. Team performance was measured by computing the process and product performance. To show the generality of the proposed model, the results were gathered after testing in every phase of the development life cycle.

Sarma *et al.* (2008) reported STC perspectives in the context of CSD. The study identifies two major STC elements, namely; social and technical artifacts with a process that measures the fit (based on existing data and instrumentation) among them. Social elements identify the team structures including direct and indirect communication. Additionally, human characteristics are also considered such as people's implicit knowledge and experience with the present condition of development. However, technical artifacts focus on the overall structure of the code, the code itself, and numerous dependencies that exist in the code components. The study emphasized that a high level of congruence among technical and social elements impacts product quality positively.

Marczak *et al.* (2009) presented a STC model to investigate the collaboration (derived from the organizational requirements) among a highly distributed cross-functional team with multiple jobs or responsibilities and consisting of developers, architects, users, customers, requirements analysts, and testers. The model computes social aspects from the underlying coordination among the members of the cross-functional team. The technical dimension consists of the tasks or work units and dependencies among them. The analysis results indicated that the fit among all of these elements helps to improve task completion time.

In the context of distributed software development (DSD), Datta (2018) recognized five STC model components: (1) developer Interaction (social element), (2) peer developers tasks that are represented through work-related links (technical element), (3) the fit that measures the coordination and communication among developers related to the work,(4) a fit strategy (based on square matrix computations) to calculate the alignment among all, and (5) an output measure as a software quality that is calculated based on defects identified in the work products.

Moreover, Cataldo *et al.* (2008) acknowledged two major elements in DSD: technical and social. The technical element comprises various aspects of technical decisions such as tasks, processes, and technology utilized in the development endeavor. The social, second element, is consists of characteristics related to the individuals in an organization (working in development) such as their norms, behaviors, and attitudes. The fit, which is based on matrices and social network analysis (SNA), identifies the relationship among the aforementioned elements that correlate with the performance towards development productivity.

Another study also reported a framework for DSD that computes the fit between two key components: the coordination structure followed by the tasks (work units), and the ongoing actual social communication activities among the team members (Valetto *et al.*, 2007). The congruence is calculated based on data extracted from the software repositories. The value of congruence directly impacts the performance of the development process and organization.

However, the model discussed by Šmite *et al.* (2012) comprises five major components: (1) system structure (technical dimension), (2) organizational structure (social dimension) that consist of geographically dispersed team members (e.g., prime contractor, direct-sub contractor, hidden sub-contractor),(3) external factors (e.g., cultural contexts, physical location, relationships, norms,

social interactions, events, ideas, people's behaviors and activities),(4) a congruence mechanism (based on task allocation strategies), and (5) the effects of congruence in terms of product completion time.

In the context of GSD, Sangwan *et al.* (2008) presented a STC framework that consists of a system structure, development team (involving project manager, product manager, integration engineer, supply manager, architect, requirements engineer, and infrastructure manager), communication and coordination mechanisms, work dependencies among team members, and a congruence measurement strategy to compute the alignment between shared artifacts. The right alignment of the system structure with a coordination mechanism results in efficient team performance.

Additionally, regarding GSD, Cataldo *et al.* (2006) referred a STC model with five elements: (1) social artifacts that indicate the coordination activities performed by individuals in different geographical locations through computer-mediated communication channels, (2) technical artifacts that comprise task dependencies among people or team based on formal organizational structures, (3) geographically dispersed teams, (4) an output that is based on task completion time which shows the efficiency of team performance, and (5) the fit (based on matrix calculation) that is the strategy to calculate the coherence among all elements.

Furthermore, Kwan *et al.*, presented a STC model for GSD that comprises: people, technical entities (including source code, a bug, a requirement, an assembled binary or a task), social relationships (identified through actual coordination) and the concept of awareness that affects the efficiency of STC measurement (Kwan *et al.*, 2011; Kwan & Damian, 2011). The proposed model calculates STC based on weighted and un-weighted congruence measurements. The computed value of congruence is used to identify the probability of software build success.

In the OSS development domain, Jiang *et al.,* defined a three-dimensional STC model contain two levels; macro and micro (Jiang *et al.*, 2012). The macro-level covers technical and social elements, such as tasks, processes, tools, techniques, expertise, people, attitudes, organizational culture, standards, and rules. On the other hand, the micro-level focuses on various factors like the knowledge, practices, beliefs, goals, skills, norms, and values of stakeholders that impact these components. The congruence among these two levels is calculated via three congruence measurements; resource-dependent, task-dependent, and knowledge-dependent congruence, which reflect the quality of the software.

Moreover, Bird *et al.* (2011) reported a STC model for OSS development that describes the relationship among the social aspect (developers, their communication structure, communication-related work dependencies, and associations among the developers), the technical aspect that covers characteristics of software (e.g., component dependencies, software complexity, and quality of software) and additional human factors (including human experience, programming expertise, and personal skills). The findings revealed that the quality of the software is associated with the match between the socio and technical aspects of development efforts.

Syeed *et al.* (2013) and Zhang *et al.* (2019) presented a similar STC strategy consisting of two main dimensions, namely; social and technical. The social dimension includes communication related activities among people who are classified according to their roles such as core members,

project leaders, and passive or active users. The second dimension is related to technical decisions such as techniques, tasks, and tasks related to dependencies. Besides the aforementioned dimensions, a few other factors are also considered effective due to the nature of OSS community members who may be in different geographical locations with diverse languages, backgrounds, cultures, and time-zones. However, the two studies presented the outputs of the proposed models in terms of software quality that directly correlates with the right alignment among both dimensions (social and technical). Table 8 provides a summary of STC components identified regarding STC in the literature.

RQ2. How did STC evolve over the time period from 2008 to 2019?

Keywords are important phrases that express the essence of a research article (Zhang *et al.*, 2014) and also highlight core areas that researchers can pursue. The investigation of the relationships among keywords leads to discovering new information about the targeted field, thus extending the boundary of knowledge in the research area. More frequent or dense keyword occurrences indicate the emerging trends and focused field of research.

To analyze the evolution in STC domain, the keywords and keyword plus were extracted from the selected pool of studies (Datta, 2018; Cataldo *et al.*, 2013; Portillo, 2013; Cataldo *et al.*, 2008; Sarma *et al.*, 2008; Colfer *et al.*, 2016; Wang *et al.*, 2018; Kwan *et al.*, 2011; Portillo-Rodríguez *et al.*, 2014; Kwan *et al.*, 2009; Šmite *et al.*, 2017; Ehrlich *et al.*, 2008; Bolici *et al.*, 2009; Šmite *et al.*, 2012; Jiang *et al.*, 2012; Gokpinar *et al.*, 2010; Valetto *et al.*, 2008; Zhang *et al.*, 2012; Bernardi *et al.*, 2012; Betz *et al.*, 2013; Marczak *et al.*, 2009; Cataldo *et al.*, 2009; Sobri *et al.*, 2017; Kwan & Damian, 2011; Calefato & Herbsleb, 2017; Cataldo *et al.*, 2008; Bird *et al.*, 2011; McLeod L, MacDonell *et al.*, 2011; Kiani *et al.*, 2013; Nguyen *et al.*, 2008; Wagstrom *et al.*, 2010; Avnet, 2016; Bettenburg, 2011; De Santana *et al.*, 2013; Suali *et al.*, 2017; Syeed *et al.*, 2013; Golzadeh, 2019; Palyart *et al.*, 2017; Zhang *et al.*, 2019; Tamburri *et al.*, 2019) for the period from 2008 to 2019. Furthermore, keywords facilitate the collection of additional information by considering various aspects like the research article, journal name, year of publication, and research field. In the present study VOS viewer (Van & Waltman, 2010) software tool was used to analyze the relationship among different keywords and additionally collected aspects. Correlation analysis of the keywords and additional aspects highlights the trends in the STC field in the eleven years. Overall, 3711 keywords were found in the selected set of articles. The records were downloaded in three batches by dividing the 2008-2019 time period in the ranges of 2008-2011, 2012-2015, and 2016-2019 (as VOS viewer does not allow more than 500 records) and were saved in .csv format.

Table 8. Summary of STC components identified in the literature

Id	Domain	STC components					
		Technical artifacts	**Social artifacts**	**Participant**	**Strategy**	**Output**	**Additional features**
Maheshwari *et al.*,(2012)	Industrial software development	Software design and planning	Interactions, regulations, and standards	Development team	Profile deviation	Team performance	-
Sarma *et al.*, (2008)		Code, code structure and code dependencies	Team direct and indirect communication	Development team	A fit process based on existing data and software instrumentation	Product quality	Implicit knowledge and experience
Marczak *et al.*,(2009)	CSD	Task or work units and work	Underlying coordination among the members of a cross-functional team	Cross-functional team (developers, architects, users, customers, requirements analysts, and testers)	Congruence derive from organizational requirements	Task completion time	-
Datta (2018)		Peer developers tasks	Developer interaction	Developers	A fit process based on square matrices computation	Software quality	-
Cataldo *et al.*, (2008)		Tasks, processes, and technology	Communication activities, norms, behaviors, and attitude	Distributed development team	A fit process based on matrices and SNA	Development productivity	-
Valetto *et al.*,(2007)	DSD	Coordination structure followed by the tasks	Actual ongoing communication activities	Team members (development team and stakeholders)	A fit strategy based on repositories data	Processes performance	
Šmite *et al.*, (2012)		System structure	Organizational structure	Team (prime contractor, direct and hidden sub-contractor)	An alignment computation based on task allocation strategies	Product completion time	External factors (cultural contexts, physical location, relationships, norms, social interaction, events, ideas, people's behaviors and activities)
Sangwan *et al.*,(2008)		System structure	Coordination and communication mechanisms	Team (project manager, product manager, integration engineer, supplier manager, architect, requirements engineer, and infrastructure manager	A mechanism to compute alignment between shared artifacts	Team performance	-
Cataldo *et al.*,(2006)	GSD	Task dependencies	Coordination activities performed through computer-mediated communication channels	Geographically disperse team	A fit strategy based on matrix calculation	Team performance and task completion time	
Kwan *et al.*, (2011); Kwan & Damian, (2011)		Source code, bug, requirement, task	Developers relationships, actual coordination	Distributed teams or team members	Congruence measurement (weighted and un-weighted measurement)	Software build success	Concept of awareness

Jiang *et al.*, (2012)		Tasks, processes, tools, techniques, expertise	People attitudes, organizational culture, standards and rule	Development team	Congruence Measurement (resource-dependent, task-dependent and knowledge-dependent measures)	Software quality	External factors (the knowledge, practices, beliefs, goals, skills, norms and values of stakeholder)
Bird *et al.*, (2011)	OSS	Components dependencies and complexity	Developers communication structure, communication related work dependencies, developers associations	Development team	Identify the relationship between socio and technical aspect by considering the human factors	Software quality	Human factors (human experience, programming expertise, and personal skills)
Syeed *et al.*,(2013) ; Zhang *et al.*,(2019)		Tasks and task related dependencies	Communication related activities	A development team consisting of core members, project leaders, passive or active users	Congruence calculation based on matrix evaluation	Software quality	Community factors (geographical locations, language, background, culture and time zones)

Table 9 provides the data drawn with the VOS viewer software tool that represents four key aspects of keywords: (1) keyword item aspect that represents the number of keywords co-occurring in different articles; (2) keywords grouped in clusters (second aspect) on the basis of their field similarity; (3) the links aspect that defines the occurrence of keywords in different articles; (4) the link strength aspect that shows the extent of co-occurrence of keyword items. The table highlights that in the 2016-2019 batch, keywords appear at a higher frequency and are grouped in 61 clusters with 23198 links and strength of 47141. The last column presents the number of keywords that reached the threshold value (a minimum of five co-occurrences).

Table 9. Keyword occurrence network aspects with respect to the time period

Keyword aspects	**2008-2011**	**2012-2015**	**2016-2019**	**Total (Meet threshold of value=5)**
Items	369	1671	1675	384
Clusters	27	59	61	19
Links	3650	23098	23198	5382
Strength	3850	35398	47141	26868

The analysis of the co-occurrence of a keyword that meets the minimum threshold is represented through a network of keywords (as shown in Figure 8). In the network, a node represents a keyword item, a line shows when two keywords are cited in a document (a thicker line represents higher cooccurrence); the size denotes the keyword frequency, and the level of similarity to the area of study is denoted by the distance between two nodes.

Figure 8 shows 19 clusters of different numbers of items and highlights the keywords with higher frequencies (co-occurrence values).

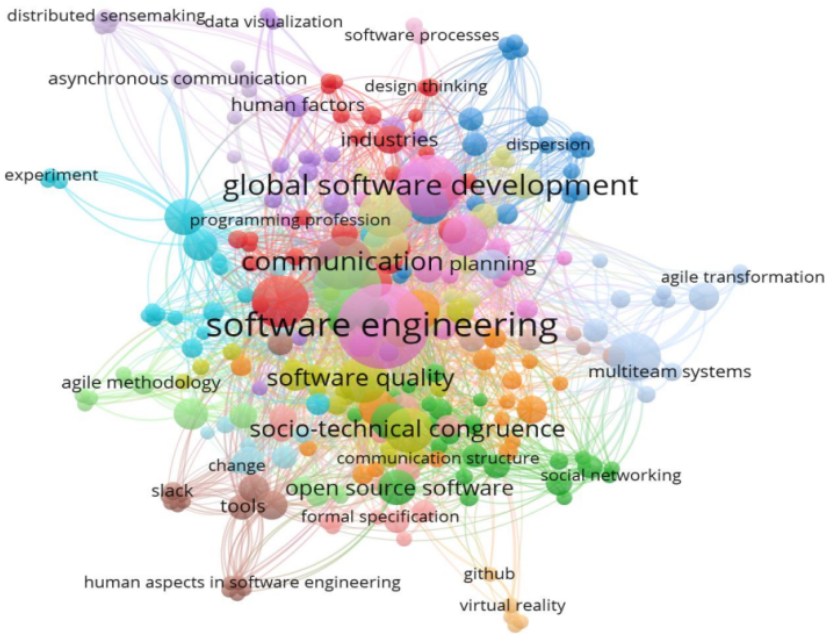

Fig. 8. Co-occurrence of keywords

Table 10 represents the top 20 co-occurring keywords with their co-occurrence values, which clearly indicates that the keywords ‘software engineering’, ‘software development’, ‘global software engineering’, ‘agile software development,’ ‘team collaboration’, ’team communication’, ’inter-team coordination’, ’software quality’ and ‘ socio-technical congruence’ denote the current focus of researchers in the field of STC.

Table 10. Top 20 co-occurring keywords

Keywords	Occurrences
Software Engineering	186
Software Development	105
Global Software Engineering	101
Agile Software Development	75
Team Collaboration	71
Team Communication	62
Inter-Team Coordination	62
Software Quality	55
Socio-Technical Congruence	53
Distributed software development	45
Interviews	44
Electronic Mail	42
Team Coordination	38
Collaborative Work	35
Data Mining	34
Open Source Software	33
Social Network Analysis	32
Global Software Engineering	32
Team Performance	27
Software Development Team	23

The trends of the top 20 keywords (as given in Table 9) for the periods 2008-2011, 2012-2015, and 2016-2019 are presented in Figure 9, with a higher occurrence of keywords in 2016-2019. In addition, it is also clear that overall the area of software engineering has received increasingly more attention from researchers between 2008 and 2019.

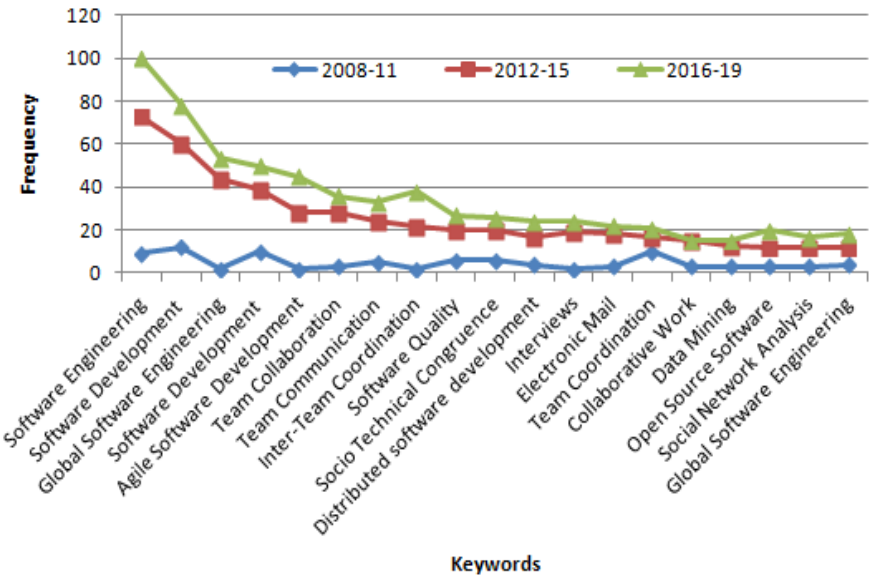

Fig. 9. Trends of top 20 co-occurring keywords from 2008 to 2019

RQ3. What is the impact of STC on software development outcomes?

Numerous studies in the literature show the relation of STC with the outcomes of software development activities. After a thorough analysis, we identified two main outcomes that indicate the effect of STC on software development in terms of improvement: (1) team or task performance and (2) software or product quality.

In the context of the team performance outcome, Maheshwari *et al.* (2012) used a profile deviation approach to calculate STC. The authors indicated that team performance will improve if the technical capabilities (infrastructure, control, and production) and social capabilities (supportiveness, conflict resolution, and communication) of underdeveloped software are aligned with each other.

Similarly, Cataldo *et al.* (2006) suggested that the task performance of developers is directly associated with the level of congruence among social (coordination activities) and technical dependencies (task dependencies). The findings of the study indicated that the higher the congruence values improve the task/team performance.

Colfer *et al.* (2016) identified a mirrored architecture that demonstrates that organizational design (e.g., mapping of tasks) is a mirror of actual ongoing communication activities. According to the results of the proposed architecture, the outcome of mirrored systems indicates an improvement in process or team performance, while the unmirrored systems reflected failure in performance.

Valetto *et al.* (2008) also assessed the impact of the STC on software development by exploring a range of alternative decisions to achieve alignment among social and technical dependencies. The study summarized the analysis with the conclusion that task or team performance is correlated with the level of congruence.

Marczak *et al.* (2009) proposed a requirements-driven collaboration approach for STC measurement. The study suggested that task completion time is one of the measures indicative of team performance. Minimal completion time shows superior performance that can be attained through a higher level of congruence among the social interactions and technical dependencies (software specification documents) of development activities.

Furthermore, Sobri *et al.* (2017) examined the impact of STC on an incremental model by analyzing the resolution time (time to resolve a bug) in software development. By applying linear regression analysis, the results indicated that congruence (among communication logs and file changed together) has significant effects on task performance.

Additionally, Sarma *et al.* (2008) presented another evidence of the correlation between STC and team performance. The study provided an analysis of the general and aligned communication effects on STC computation. To compute the value of STC, the authors utilized data (communication data and task dependencies) extracted from software repositories. The results showed that a high level of congruence boost team performance.

On the other hand, in terms of software quality improvement, Datta (2018) analyzed the correlation between communication and the work structure of an organization to analyze the impact of STC on software development outcomes. To measure the value of STC, the proposed study identified the relationship between social (communication activities) and technical (files) artifacts. The congruence value among both artifacts indicated a lower number of defects in the development activities, which demonstrates improvement in software quality.

Suali *et al.* (2017) also highlighted a method of evaluating the relation of STC with software development, specifically the STC impact on software quality. The value of STC is computed by comparing the coordination needs (source code artifacts) with actually performed coordination. The results were in line with the Datta technique (Datta, 2018), indicating that a lower number of defects enhance software quality.

Similarly, Valetto *et al.* (2007) correlated the impact of STC with the number of defects identified during development. For STC measurement, the proposed approach was used to investigate software repositories containing data related to communication activities and the source code utilized during development. The results illustrated that the number of defects is greater when the level of congruence is low.

However, the study by Cataldo *et al.* (2013) revealed that the chances of software failure increase when the gap between the actual and required coordination is maximized. According to the analysis, a high level of congruence (no gap or smaller gap between the actual and required coordination) leads to enhanced productivity as well as software quality.

The framework presented by Cataldo *et al.* (2008) suggests that the value of STC is an indicator of the way to perceive the unsatisfactory level of coordination needs affect the productivity of

development activities. The findings showed that developers who coordinate their tasks with the group of workers concerned achieve progresses in development productivity.

Furthermore, Kwan *et al.* (2011) presented a combination of weighted and un-weighted congruence framework extended with the concept of awareness. The proposed framework measures the association between coordination activities and coordination needs to find the success probability of software builds. The results revealed that a high level of congruence leads to probable successful builds (an indication of better software quality).

Portillo-Rodríguez *et al.* (2014) discussed an agent-based STC architecture and tool that use agent properties to manage development activities. A case study was carried out using the proposed tool with new agent features to measure the impact of STC on software development. The outcomes of the case study denoted improvement in productivity and software quality through achieving an effective level of coordination.

Bettenburg (2011) identified the close association between the quality of software and the level of congruence (measured based on collaboration between stakeholders). The findings suggested that elevated levels of collaboration among stakeholders positively affect the performance of the development community and the code quality.

More recently, Zhang *et al.* (2019) presented a correlation of STC with moderate bug proneness in OSS projects. The study proposed a model to compute the value of STC by investigating the source code and issue tracking systems. It was determined that a lower degree of STC presents higher numbers of bugs, which indicates poor software quality. Table 11 provides a summary of STC impacts on software development outcomes.

Table 11. Summary of STC impacts on software development outcomes

Id	Approach	Social aspect	Technical aspect	Performance impacts
Maheshwari *et al.,* (2012)	Profile deviation	Supportiveness, conflict resolution, and communication	Infrastructure, control, and production capability	Improved team performance
Cataldo *et al.,* (2006)	Dependency relationship analysis	Coordination activities	Task dependencies	Improved task /team performance
Colfer *et al.*, (2016)	Organizational ties And technical dependencies analysis	Collocation, employment relations, and communication channels	Task dependencies	Improved process or team performance
Valetto *et al.,* (2008)	Analysis of software graph and social network	Communication and coordination activities	Task and task dependencies	Improved task, and team performance, reduced amount of rework
Marczak *et al.,* (2009)	Requirement driven collaboration analysis	Communication activities	Requirement specification documents	Improvement in people/team performance
Sobri *et al.*,(2017)	Analysis of actual and required coordination in an incremental model	Communication logs(mailing list)	Files dependency	Improved task performance
Sarma *et al.*, (2008)	Analysis of aligned and general communication among individuals	Communication activities	Task dependencies	Reduces bug resolution time (software quality)
Datta, (2018)	Analysis of communication and work structure	Communication activities	Files dependency	Improved software quality
Suali *et al.,* (2017)	Analysis of the match between coordination needs and actual coordination	Coordination interactions	Source code	Improved software quality
Valetto *et al.,*(2007)	Investigation of software repositories	Communication activities	Source code, traceability relationships	Reduces the number of defects(software quality)

Cataldo *et al.*, (2013)	Analysis of the match between coordination needs and actual coordination	Communication activities (email)	Source code	Improved development productivity
Cataldo *et al.*, (2008)	Analysis of coordination needs and coordinating actions	Coordination activities	Tasks (source code)	Effective product management (software quality)
Kwan *et al.*, (2011)	A hybrid model with the awareness concept	Email	Source code	Improved build success rate(software quality)
Portillo-Rodríguez *et al.*, (2014)	Agent-based analysis	Communication interactions	Source files	Improved productivity and software quality
Bettenburg,(2011)	Analysis of stakeholder coordination	Stakeholder's communication logs	Source code entities	Improved software quality
Zhang *et al.,* (2019)	Analysis of the congruence relation with bug proneness	Email, bug, and issue tracking repositories	Source code	Improved software quality

RQ4. What are the consequences or effects of non-congruence?

To address the impacts of non-congruence, several existing studies that highlight the effects of non-congruence on software development are investigated. The identified non-congruence effects are analyzed in terms of software failure, project delay, frequent code change requests, bugs in the development process, poor team performance, and lower product quality.

Evidence of non-congruence effects related to software failure was addressed in Cataldo *et al.* (2009). The research discovered that a gap in the identification of socio- or technical dependencies causes ultimate software failure. In (Šmite & Galviņa, 2012; Syeed *et al.* 2013) it was reported that low STC levels introduce issues in coordination and thus integration failure for components developed by distributed teams.

In the context of project delay, Marczak *et al.* (2009) highlighted the problem of low coordination from a literature analysis as a gap in coordination that leads to incremented resolution time of team activities, thus producing project timeline delays.

Moreover, in Ehrlich *et al.* (2008) the correlation between gaps in congruence and code changes was described. Gaps produce low productivity and an increased number of code changes.

Regarding a team communication network, another issue of non-congruence was highlighted in (Bernardi *et al.*, 2012): the ratio of bugs is higher for committers who coordinate less than other committers. In Zhang *et al.* (2012), the coordination patterns of file edits were analyzed, and it was found that inadequate links among developers lead to bugs in the files.

The negative effects of low STC on team performance within an organization were addressed in (Sosa *et al.*, 2004). The results of non-congruence were discussed in (Betz *et al.*, 2013; Sarma *et al.*, 2008) in terms of poor performance, low quality, and additional costs due to coordination overhead. Gokpinar *et al.* (2010) identified the negative result of mismatches among derived and actual coordination needs as poor complex system quality. Table 12 presents the effects of non-congruence in various scenarios.

Table 12. Consequences of non-congruence

Id	Consequences of non-congruence
Cataldo *et al.*, (2009)	Software failure
Šmite & Galviņa, (2012); Syeed *et al.,* (2013)	Component integration failure
Marczak *et al.*, (2009)	• Incremented resolution time • Delayed project timeline
Ehrlich *et al.*, (2008)	• Low productivity • Increased number of code changes
Bernardi *et al.,* (2012); Zhang *et al.* (2012)	Bugs in files
Sosa *et al.,* (2004); Betz *et al.*, (2013); Sarma *et al.*, (2008); Gokpinar *et al.,* (2010)	• Poor performance • Low quality • Raised project cost

RQ5. What socio-technical data sources are used in literature for STC measurement?

An in-depth analysis of the selected studies obtained from various data sources used to extract socio-technical knowledge related to software development was carried out to facilitate STC computation. Seven types of data sources were found: email data, chat history, VCS, MR, documentation, qualitative data repository, and miscellaneous data. The miscellaneous category represents a collection of general data gathered in different ways. A summary of STC-related data sources and features are given in Table 13. The detail of each data source is as follows:

- *Email Data*

Several authors (Maheshwari *et al.*, 2012; Datta, 2018; Sangwan & Ros, 2008; Valetto *et al.*, 2007; Portillo-Rodríguez *et al.*, 2014; Bolici *et al.*, 2009; Bernardi *et al.*, 2012; Sobri *et al.*, 2017; Kwan & Damian, 2011; Wagstrom *et al.*, 2010; Suali *et al.*, 2017) indicated that email repositories such as email lists and recoded communication are a source of communication between users, developers, stakeholders, and other team members. The studies highlighted that email data is usually utilized more as a data source during software development.

- *Chat History*

A few studies (Cataldo *et al.*, 2006; Cataldo *et al.*, 2013; Portillo-Rodríguez *et al.*, 2014) discussed the Internet Relay Chat (IRC) or integrated chats application as a data source that provides information about the coordination and communication between team members regarding development activities.

- *Version Control System (VCS)*

Information related to source code, a historical record of source code including revisions, change logs, and changed authorships with metadata can be found in VCS. It also stores communication data related to code changes. Several studies (Maheshwari *et al.*, 2012; Sangwan & Ros, 2008; Cataldo *et al.*, 2006; Cataldo *et al.*, 2013; Cataldo *et al.*, 2008; Valetto *et al.*, 2007; Šmite *et al.*, 2012; Jiang *et al.*, 2012; Sobri *et al.*, 2017; Wagstrom *et al.*, 2010; Bettenburg, 2011; Suali *et al.*, 2017; Syeed *et al.*, 2013; Golzadeh, 2019; Palyart *et al.*, 2017; Zhang *et al.*, 2019; Tamburri *et al.*, 2019) refer to VCS as a major source for extracting technical information via file dependency networks (file versus developer network).

- *Modification Request (MR)*

MR is a data repository that stores information related to bugs, their tracking, and task-related tracking or modification requests of the source code files. Many studies (Datta, 2018; Cataldo *et al.*, 2006; Cataldo *et al.*, 2013; Cataldo *et al.*, 2008; Kwan *et al.*, 2011; Portillo-Rodríguez *et al.*, 2014; Ehrlich *et al.*, 2008; Bolici *et al.*, 2009; Bernardi *et al.*, 2012; Bettenburg, 2011; Golzadeh, 2019; Palyart *et al.*, 2017) mention MR as a data source to extract information, such as defect density, communication-related to bugs, developer versus task network or coordination requirements in the process of STC computation.

- *Documentation*

Documents about different stages of software development like project plans, requirement specifications, software design, implementation specifications, issue reports, and text documents are included in this data source. Some authors (Sangwan & Ros, 2008; Šmite *et al.*, 2012; Marczak *et al.*, 2009) addressed documentation utilization for inspection, tracing the task dependency, tracking the development dependency, and identifying coordination requirements.

- *Qualitative Data*

Data collected through surveys, questionnaires, or interviews fall in this category. In numerous studies (Valetto *et al.*, 2007; Ehrlich *et al.*, 2008; Bolici *et al.*, Šmite *et al.*, 2012; Jiang *et al.*, 2012; Marczak *et al.*, 2009; Tamburri *et al.*, 2019) qualitative data sources are applied for hypothesis formulation and validating results.

- *Miscellaneous*

Data obtained via informal ways of communication (e.g., meetings, voice calls, discussion forums), integrated tools, equipment specifications, skills, and other methods of interactions (e.g., reachability, accessibility, and clustering) are included in this category (Datta, 2018; Sangwan & Ros, 2008; Portillo-Rodríguez *et al.*, 2014; Ehrlich *et al.*, 2008; Bolici *et al.*, 2009; Šmite & Galviņa, 2012; Jiang *et al.*, 2012; Kwan & Damian, 2011).

RQ6. What techniques exist to measure STC?

STC is a way of measuring and enhancing the development of team performance at both global and local levels. To measure STC efficiently, existing STC techniques rely on different tools, such as email, forums, instant messaging, or software repositories. The required information is then extracted automatically from these tools. However, manual methods of collecting information are also adopted, such as interviews or surveys. Such automatic or manual STC tools not only help to measure STC itself but also provide ways to detect and solve problems that may arise during coordination.

From the set of STC measurement studies, it is observed that congruence analyses have mostly been applied to the third phase of development (coding phase) of a project. As a result, the source code file serves as the technical entity in most published STC works. Moreover, it is observed that social interaction tools (i.e. distribution lists, comments, and Internet Relay Chat (IRC) communication) are most commonly used for congruence analysis to calculate the actual coordination measure.

Table 13. Summary of STC related data sources and features

Paper Id	STC data source	Features of data sources
Maheshwari *et al.*, (2012); Datta, (2018); Sangwan & Ros, (2008); Valetto *et al.*, (2007); Portillo-Rodríguez *et al.*, (2014); Bolici *et al.*, (2009); Bernardi *et al.*, (2012); Sobri *et al.*, (2017); Kwan & Damian,(2011); Wagstrom *et al.*, (2010); Suali *et al.*, (2017)	Email data	Email lists, recoded communication
Cataldo *et al.*,(2006); Cataldo *et al.*, (2013); Portillo-Rodríguez *et al.*, (2014)	Chat history	IRC, integrated chats applications
Maheshwari *et al.*, (2012); Sangwan & Ros, (2008); Cataldo *et al.*, (2006); Cataldo *et al.*, (2013); Cataldo *et al.*, (2008); Valetto *et al.*, (2007); Šmite *et al.*, (2012); Jiang *et al.*, (2012); Sobri *et al.*, (2017); Wagstrom *et al.*, (2010); Bettenburg, (2011); Suali *et al.*, (2017); Syeed *et al.*, (2013); Golzadeh, (2019); Palyart *et al.*, (2017); Zhang *et al.*, 2019; Tamburri *et al.*, (2019)	Version control system (VCS)	Source code, a historical record about source code, code revisions, change logs, changed authorships code metadata
Datta, (2018); Cataldo *et al.*, (2006); Cataldo *et al.*, (2013); Cataldo *et al.*,(2008); Kwan *et al.*, (2011); Portillo-Rodríguez *et al.*, (2014); Ehrlich *et al.*,(2008); Bolici *et al.*, (2009); Bernardi *et al.*, (2012); Bettenburg, (2011); Golzadeh, (2019); Palyart *et al.*, (2017)	Modification request (MR)	Bugs, task and bug tracking, modification requests
Sangwan & Ros, (2008); Šmite *et al.*,(2012); Marczak *et al.*, (2009)	Documentation	Software plans, requirement specification, design, implementation specifications, problem reports, text documents
Valetto *et al.*, (2007);Ehrlich *et al.*, (2008); Bolici *et al.*, Šmite *et al.*, (2012); Jiang *et al.*, 2012; Marczak *et al.*, (2009); Tamburri *et al.*, (2019)	Qualitative data	Responses from surveys, questionnaires, and interviews
Datta, (2018); Sangwan & Ros, (2008); Portillo-Rodríguez *et al.*, (2014); Ehrlich *et al.*, (2008); Bolici *et al.*, (2009); Šmite & Galviņa, (2012); Jiang *et al.*, (2012); Kwan & Damian, (2011)	Miscellaneous	Record of voice calls, discussion forums, integrated tools data, equipment specification, skills

Various approaches to STC have been proposed in the literature. In this study, these approaches are classified into two broad categories. The first type consists of approaches in which matrices are used to represent coordination needs and technical dependencies. The second type of STC approach employs social network analysis as a means of computing STC.

Cataldo *et al.*, (Cataldo *et al.,* 2008; Cataldo *et al.*, 2006) used matrices to measure STC. In the proposed methodology, four different coordination measures were calculated from the task dependencies and actual coordination. One of these measures is structural congruence, whereby coordination is calculated among two members of the same team. The second measure is geographical congruence, which is calculated when both team members are functioning at a similar physical location. Modification request (MR) communication congruence is the third category of measurement, which entails measuring when two team members are commenting on the same modification request. The final coordination measure is IRC communication congruence, which is calculated manually from IRC logs when two team members refer to the same MR.

Based on the model proposed by (Cataldo *et al.*, 2006), Kwan *et al.*, (Kwan *et al.*, 2009; Kwan *et al.*, 2011) presented an improved STC measurement to overcome the limitations of Cataldo *et al.*'s work. The authors added a new measurement to their model, namely, weighted congruence. The model assigns weights to the dependencies that exist among tasks, as well as those among people and tasks. These weights are measured by calculating the ratio of task dependencies on other tasks to the total number of dependencies among tasks. The task-to-people ratio weight is calculated based on the people's total working hours on a particular task. The model investigated the communication network to obtain actual coordination by assigning weights to ongoing

communication among team members or tasks. The proposed weighted congruence model successfully detects coordination gaps and suggests the key coordination tasks of the highest priority that need to be managed for better performance.

Another vein of STC was presented by Valetto *et al.* (2007). They proposed an STC measurement model based on social network analysis (SNA). The proposed model measures congruence based on the relationships among software artifacts, trust among team members, and the size of their contribution. The relationships among software artifacts are analyzed through a network of directed and undirected graphs. Undirected graphs illustrate information regarding communication interaction, whereas work relationships and dependencies among software artifacts are represented through directed arcs. The arcs are assigned weights according to the frequency of modifications made by a team member on the same artifact or dependencies among source files. To calculate the congruence, the ratio of the relationships among software artifacts is measured using the proposed algorithm.

In another study, Gokpinar *et al.* (2010) also used SNA for congruence measurement. The authors created the product architecture network, which consists of nodes and arcs that represent subsystems and links or dependencies among subsystems, respectively. To obtain actual coordination, the authors created another network called an organizational coordination network that illustrates the dependencies or communication among engineers and subsystems. To calculate congruence, the proposed model compares both networks and identifies the coordination gap. The model reports this gap as a coordination deficit.

More recently, Zhang *et al.* (2019) proposed a STC model with an additional concept of missing a developer's link. The model computes the value of STC at the building level. The effectiveness of the proposed model was determined through continuous prediction of defects using 10 GitHub projects. The findings reveal that the addition of STC helps to predict defects at the building level. Furthermore, the study highlighted the significance of missing developer links over STC values. However, the value of congruence is calculated by taking the ratio of coordination requirements over actual coordination.

Moreover, Kwan *et al.* (2011) presented an improved STC model by introducing the concept of awareness in STC measurement. An empirical study was performed on a ship project to measure STC with awareness via interviews, direct observations, and a questionnaire. The main purpose was to observe the flow of awareness information among a globally distributed team. After a detailed analysis, the authors found that the team members are satisfied with simple awareness approaches, such as email, chat, or meetings. The proposed model also noted the role of a broker (an experienced team member) in filling the coordination gap among team members. A major finding pertains to an aggregated STC measurement that can accommodate multiple relationships to satisfy social as well as technical dependencies.

A new three-dimensional STC measurement model was proposed (Jiang *et al.*, 2012). The researchers argued that existing STC research mainly focuses on measuring the congruence between task dependency and developer coordination, which represents the measure of fit of developer interaction concerning task dependencies. However, coordination requirements are not only based on task dependencies but also on other factors, such as sharing of knowledge, resources,

and expertise. Therefore, the model adopts two additional concepts: knowledge and resource dependent congruence with existing congruence measurement to obtain better results.

Wagstrom *et al.* (2010) proposed an individualized STC (ISTC) model based on Cataldo *et al.*'s (Cataldo *et al.*, 2008; Cataldo *et al.*, 2006) work. For measurement, the proposed model creates an ISTC matrix from the contribution of each individual person. Congruence is calculated by taking the ratio of the difference between the expected coordination needs of an individual and the coordination requirements. The proposed model can identify the difference among simple, normal, or specific communication that should be conceded to mitigate the task dependencies.

Portillo-Rodríguez *et al.* (2014) proposed a multiagent STC model for GSD. The proposed model utilizes the concept of Kwan *et al.*'s model (Kwan *et al.*, 2009) and some additional factors related to environmental needs. It is important to consider environmental factors because they influence the coordination among globally distributed team members. Unlike Kwan's work, the STC of individual users is calculated using three types of weights: socio-cultural distance (SCD), temporal distance (TD), and geographical distance (GD). With these additional factors (i.e., weights), task priorities are incorporated in the proposed model. It is important to note that Portillo Rodríguez's model can be incorporated in every phase of the project lifecycle.

In (Avnet, 2016), a network-based approach was presented to measure team coordination. The authors used this measurement to calculate the shared cognition of an engineering design team. The model contained a design structure matrix (DSM) from technical data flow, whereas information on the actual interaction was collected through a survey of 10 integrated concurrent engineering (ICE) design teams. A comparison was then performed between DSM and the reported data to compute the STC measure. With the STC measurement and pair-wise shared mental models, the proposed technique identified that both team coordination and design products are related to shared cognition.

Moreover, Zhang *et al.* (2019) proposed an STC model based on Cataldo *et al.*'s framework for OSS projects. The proposed model embeds some adjustments in a baseline model according to the OSS environment to measure STC on the file level. The proposed work added a new derivative matrix, Missing Developer Links (MDL), which helps to measure the coordination breakdown among the OSS developers. The values of STC and MDL were computed from the file and developer networks, respectively. The method calculates the relationship among the STC/MDL and bug proneness that correlate with software quality. The effectiveness of the proposed method was validated through an empirical study on five OSS projects. The results show improvement in bug prediction, which enhances software quality.

To summarize, various STC measuring techniques are cross-compared in this study using five features: technique name, major characteristics, strategy, metrics, result validation, and measurement dimensions, as given in Table 14.

RQ7. Which factors influence STC measurement?

According to various organization forms, such as organizational distributed development, collaborated development, global distribution, product development, and open-source, the literature highlights numerous factors that influence STC measurement. In an organization, the

level of coordination among teams varies significantly according to three major factors: the degree of module coupling, organizational architecture changes, and nonfunctional requirements. However, many other factors influence organizational technical decisions, such as geographic distance, language barriers, socio-cultural differences, domain expertise, etc.

Table 14. Summary of STC measurement techniques

Id	Name	Characteristics	Strategy	Metric	Empirically Validated	Measurement Dimensions
Cataldo *et al.*, (2008); Cataldo *et al.*, (2006)	STC framework	First congruence model based on Conway's law.	Matrix	Un-weighted	Yes	• Structural congruence • Modification request (MR) • IRC logs
Kwan *et al.*, (2009)	Software build success STC model	An improved weighted congruence model based on Cataldo *et al.*'s (2008) work.	Matrix	Weighted	Yes	• Task dependencies • Communication pattern • Team working hours
Valetto *et al.*, (2007)	STC model based on software repositories	A model based on weighted social network analysis extracted from software repositories.	SNA	Weighted	Yes	• Software repositories
Gokpinar *et al.*, (2010)	Product quality STC model	SNA-based congruence measurement. Two networks are analyzed for calculation: product architecture network, and organizational coordination network.	SNA	Weighted	Yes	• Product architecture network • Organizational coordination network • Communication through a distributed list
Kwan *et al.*,(2011)	STC model with awareness	An improved STC model with a concept of awareness.	Matrix with team awareness	Weighted	Yes	• Interviews • Direct observations • Questionnaire
Jiang *et al.*, (2012)	Team performance STC model	An improved three-dimensional STC measurement.	Matrix	Un-weighted	Yes	• Task dependency • Knowledge dependency • Resource dependency
Wagstrom *et al.*, (2010)	Individualized STC model	An individualized STC (ISTC) model based on Cataldo *et al.*'s (Cataldo *et al.*, 2008; Cataldo *et al.*, 2006) studies.	Matrix	Weighted	Yes	ISTC matrix
Portillo-Rodríguez *et al.*, (2014)	Agents-based STC model	A multi agent STC model based on Kwan *et al.* (Kwan *et al.*, 2009) for GSD.	Matrix with agent properties	Weighted	Yes	• Socio-cultural distance • Temporal distance • Geographical distance
Avnet, (2016)	SNA-based STC model	An improved network-based approach to measure congruence and shared cognition among an engineering design team.	SNA and matrix	Weighted	Yes	• Design structure matrix • Survey integrated concurrent engineering
Zhang *et al.*, (2019)	File-Level STC model	A model for OSS development for bug prediction on file-level.	SNA and matrix	Weighted	Yes	• Files • MDL

An organization will achieve high congruence when its coordination capabilities match or surpass the coordination requirements. According to Cataldo *et al.* (2008) congruence is defined as the match between the organizational design and abilities to carry out a task. The two main factors that influence congruence calculation are organizational temporal dependencies (dependencies among the team and assigned tasks) and organizational structure (the method of coordination and communication).

However, Sobri *et al.* (2017) highlighted numerous factors from the literature that contribute to successful coordination among team members within an organization. In terms of team coordination, Kwan *et al.* (2011) identified many explicit (i.e., email, chat, meetings) and implicit communication mechanisms (i.e., listening or learning from other work) that contribute to strengthening the coordination among distributed teams. To make coordination more successful, Calefato *et al.* (2012) added two more factors to those of (Kwan *et al.*, 2011): managers' contributions and instilled trust among team members. According to Cataldo *et al.* (2008), the success of project development is highly dependent on the structural patterns of communication, which affect project performance. Other studies (Kwan *et al.*, 2011; Kiani *et al.*, 2013; Nguyen *et al.*, 2008) presented yet another factor, awareness among team members, which influences the mechanism of coordination.

Moreover, distributed software development (Jiang *et al.*, 2012) classified STC influencing factors in two categories: macro-level and micro-level. Macro-level factors consist of team values, practices, knowledge, expertise, beliefs, behaviors, norms, and aims of the stakeholders. Micro level factors affect both the technical and social aspects of software development. Technical components comprise project tasks, knowledge, mechanisms, techniques, and tools. However, social aspects comprise components related to people, such as norms, culture, behavior, and team attitude. For successful project development, it is necessary to attain congruence at both levels.

Dynamic social network analysis (SNA) is one means of calculating STC in collaborated development. In Datta (2018), an SNA-based STC measurement technique was addressed that is significantly affected by one factor, which is the relationship among organizational entities (e.g., tasks, resources, and people). In the literature, additional factors identified are related to task elements, such as the attributes of developers involved in the task, lines of code to be written, task assignment, and task priority, which can affect task completion time and the resolution of modification requests. In the context of collaborative development, (Bird *et al.*, 2011) identified critical human factors that affect the process of software development. The human factors identified (for instance, the number of team members working on a file and the number who left the organization before project completion) are linked to software quality and team performance.

Based on the work of (Kwan *et al.*, 2011) Portillo Rodríguez *et al.* (2014) highlighted a range of factors that can influence the coordination mechanism in global software development. Socio cultural distance (SCD) is one of the factors identified and is calculated based on the native countries in the team included in the development process. Another factor is Temporal Distance (TD), which is based on the time difference between the locations of the working team members. To attain a sufficient level of coordination, it is assumed that more communication is necessary among team members from different time zones. To determine the STC for each user in a team, these factors are assigned some weights based on cultural and temporal distance.

In terms of product development, Cataldo *et al.* (2013) identified two significant contextual factors: product and process maturity. These impact the relationship among software dependencies, coordination needs, and product quality.

Moreover, McLeod *et al.* (2011) recognized three classes of factors that influence STC measurement. The first class consists of people and their actions, such as developers, top

management, project team, external agents, users, and social interaction. The other class is project content and involves project goals and objectives, scope, resources, characteristics, and technology. The last class consists of factors related to the development process, for instance, requirement determination, utilization of a standard methodology, project management, user participation, user training, and change management.

In the context of OSS development, Tamburri *et al.* (2019) presented the relationship between STC and open-source community smells that define the suboptimal organizational patterns and socio-technical characteristics. The fewer the community smells, the higher the STC value that improves software quality. A complete list of factors affecting STC measurement is given in Table 15

Table 15. Summary of factors influencing STC measurement

Id	Context	Factors	Results
Cataldo *et al.*, (2008) Kwan *et al.*,(2011); Kiani *et al.*, (2013); Nguyen *et al.*, (2008)		• Temporal dependencies • Organizational structure • Explicit and implicit communication • Awareness	Improved team coordination and communication • Strengthening the coordination mechanism • Contributing to the mechanism of successful coordination
Calefato *et al.*,(2012)	Organizational distributed development	• Trust • Team leadership	Makes coordination successful
Cataldo *et al.*,(2008)		Structural patterns of communication	Improved project performance and outcome as successful project development.
Jiang *et al.*, (2012)		• Technical aspects o Tasks, knowledge, mechanisms, techniques, and tools • Social aspects o Norms, culture, behaviors, and attitudes	Successful project development
Datta, (2018)		Relationship among organizational entities	• Identified dependencies among tasks, resources, and individuals
Bird *et al.*, (2011)	Collaborative development	Human factors	• Identified needs for coordination requirements • Improved software development mechanism • Reduced software failure
Portillo- Rodríguez *et al.*,(2014)	GSD	• Socio-cultural distance (SCD) • Temporal distance (TD)	Improved coordination mechanism in GSD
Cataldo *et al.*, (2013)		• Product maturity • Process maturity	• Identified dependencies among software components, coordination requirements • High software quality
McLeod *et al.*, (2011)	Product development	• Product, and people and their actions o Developers, top management, project team, external agents, users, and social interaction • Project content o Project goals and objectives, scope, resources, characteristics, and technology • Development process o Requirement determination, utilization of a standard methodology, project management, user participation, user training, and change management	• Improved team performance • High quality project
Tamburri *et al.*, (2019)	OSS	• Sub-optimal organizational patterns • Socio-technical characteristics	• Improved Organizational Quality

3. Discussion

3.1 Principal findings

From a detailed analysis of the selected study pool in the context of the proposed research questions, several aspects of STC are recognized. For instance, each STC study provides its own conceptualization of STC components according to the development environment (as discussed in RQ1). Most studies focus on the domains of DSD, GSD, and OSS development. However, the first operational STC model was presented by Cataldo *et al.*, (2006) for organizational distributed software development. Upon investigating studies to answer the second RQ, it was found that STC measurements have been more geared towards software engineering and global software development. The topmost co-occurring keywords prevalent in the field of STC are; team collaboration, communication, and inter-team coordination. Most STC-related studies present quantitative features of STC. The quantitative calculation of STC enables the analysis and proposal of scores for the social and technical concerns of a project much more easily.

Several studies were identified that illustrate the importance of STC in improving team performance and project quality (RQ3). From a practitioner's point of view, it was identified that STC measurement better assists managers with inspecting and controlling team members' activities, which in turn enhances team performance and software quality.

In contrast to the benefits, the existence of non-congruence presents negative impacts on software development like software failure, poor task performance, and low software quality (as presented in RQ4).

Through an analysis of the selected studies, seven major types of data sources were explored; email data, chat history, VCS, MR, documentation, qualitative data, and a miscellaneous data repository, which were used to extract social and technical information for STC calculation (RQ5).

The present SLR classifies STC techniques into two groups: matrix-based STC techniques and social network analysis (SNA) techniques. These techniques can be further categorized as weighted and unweighted measurements (as highlighted in RQ6). A notable aspect is that, thus far, only weighted measurement methods have been exploited in SNA approaches. Thus, SNA with unweighted measurement still needs to be explored.

Finally, this study summarizes the major factors affecting STC measurement and consequently on the results of software development (team performance and software quality). In addition, numerous key factors (with various sub-factors) influencing STC calculation were determined. The factors relate to different fields, such as organizational, collaborative global software development, product development, and open-source projects (RQ7). Figure 10 presents the taxonomy of RQs and SLR findings.

3.2 Future implications

After analyzing the selected studies, the following research directions were identified. Existing STC studies highlight the benefits and importance of STC, but only one study was found to primarily discuss the risk of overwhelming STC. Risk is crucial to be investigated and explored

further to ensure the success of any projects. Hence, there is an urgent need to motivate more work in this area.

The analysis of existing STC measurement techniques demonstrates that existing techniques utilized weighted and unweighted STC measurement metrics. However, STC calculation based on the unweighted SNA mechanism still needs in its infancy. Furthermore, it seems obvious that STC measurement is mainly employed in the development phase of the project lifecycle. Few research works discuss STC application in other phases (e.g., requirement, testing, or all phases) of the project lifecycle. It was observed that STC is a multidimensional technique covering all levels of the projects lifecycle. Thus, more research evidence is still required to verify the significance of using STC in different phases of software development. Subsequently, using STC techniques to identified congruence gaps at different phases of development may be a prospective area of research in the future. Likewise, the literature shows that STC measurement techniques are influenced by numerous factors, which impels the need to explore and investigate more in this area.

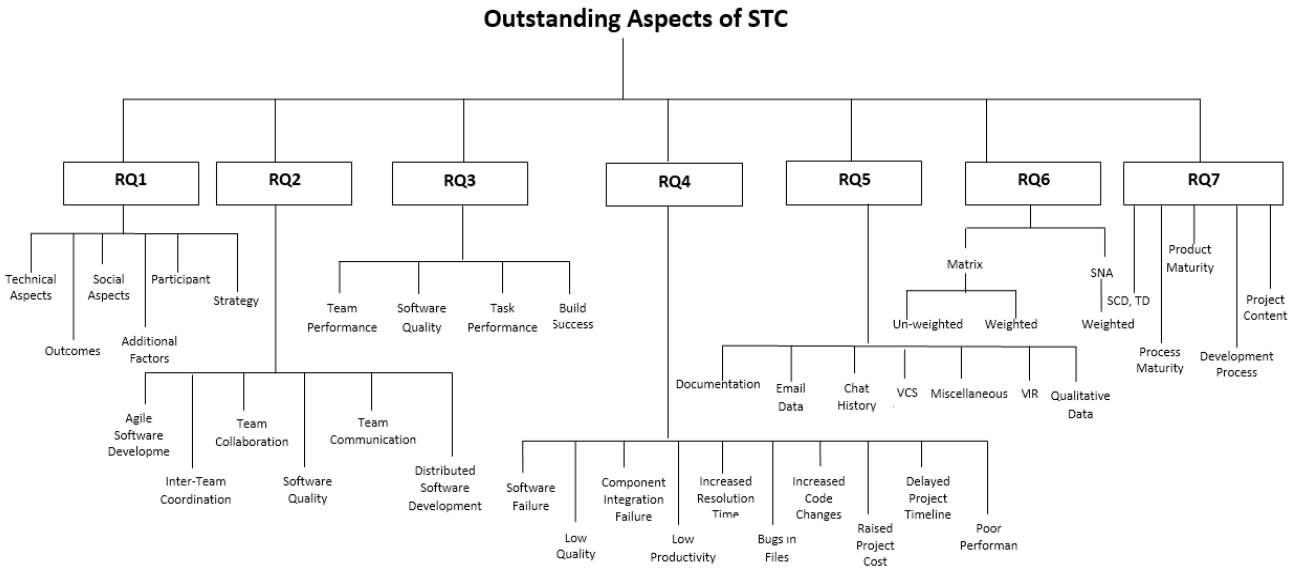

Fig. 10. Taxonomy of RQs and findings

4. Threats to validity

To avoid future replications and other aspects of this SLR, some limitations and concerns must be considered. For this SLR, validity threats were classified into three categories: internal, external, and conclusion validity.

Internal validity concerns may include biased decisions that may occur during study selection and data extraction. For instance, studies that do not provide flawless result descriptions may not be included in the selected paper pool. To mitigate this validity threat, the authors selected and

synthesized the studies. All authors cross-checked the results and discussed the outcomes until the final consensus was reached.

The external validity threat is apprehensive to SLR results and indicates to which degree the review topic is represented by the primary studies selected. This threat was reduced by ensuring that all potential studies related to the research area were covered. To this end, an extensive search was performed in four digital data sources using different related queries. Furthermore, to mitigate any limitations, iterative search criteria were employed, and the search results were validated via discussion among all authors. All backward references in the selected studies were also searched manually to ensure the inclusion of all relevant studies.

The last validity threat, conclusion validity, is related to the exclusion of some studies that should have been included in this SLR. This threat was mitigated by performing a systematic selection procedure. The selection process was performed based on the defined inclusion and exclusion criteria that were carefully designed after consensus and discussion by the co-authors to reduce the risk associated with the exclusion of relevant studies.

5. Conclusion

In this study, an SLR was conducted about STC key aspects and new trends. More specifically, the main aim of this SLR was to comprehend the concept and efficiency of the STC mechanism towards its impact on software development.

This SLR identified 46 studies relevant to the seven research questions from four large data sources. After a detailed analysis of the selected studies, the study has identified six major components of a STC model according to the development context, shown significant contribution in different domains of software engineering such as GSD, Agile development, and team performance, and identified the impact of STC in terms of four performance metrics (i.e.,). Additionally, the study revealed different disasters due to lack of congruence in software development and classified STC data sources into seven major categories depending on the resources used to extract technical and social information. Furthermore, it is concluded that each existing STC technique is an adaptation of either Cataldo *et al.* (2008) or Kwan *et al.*'s (2011) work. This study classified these techniques into 2 groups: (1) matrix-based calculation based on unweighted and weighted measurement (2) social network analysis based on weighted measurement. However, the existing STC measurements face challenges due to numerous influencing factors.

Many studies have discussed different aspects and prominent concepts about STC. However, some limitations have been identified from the thorough analysis of selected studies. This SLR led to the proposal of four investigation directions: (1) identifying the risks of overwhelming STC, (2) determine an STC measurement to overcome the communication gaps identified using unweighted SNA technique, (3) use STC performance measures in different and all phases of the project lifecycle, and (4) explore STC influencing factors.

The utility of this SLR for software developers, managers, and researchers lies in the recognition of core aspects of STC and gaps in the existing literature. We expect the findings of this SLR to enhance knowledge about STC towards successful and less error-prone software development.

ACKNOWLEDGMENTS

We wish to acknowledge the Universiti Malaya grant GPF097B-2020 for partially supporting this research and our colleague Dr. Khubaib Amjad Alam (KAA) for helping with the review process.

References

Abbasi, F. A., Burdi, A., KHAN, R. S., & NAQVI, S. H. F. (2019). Requirement Engineering Challenges in Distributed Software Development. Sindh University Research Journal -Science Series. **51**. 465-. 10.26692/surj/2019.09.74.

Aljazzaf, Z. M. (2015). Modelling and measuring the quality of online services. Kuwait Journal of Science, **42**(3).

Alqarni, T. & Qureshi, M. R. (2019). A unified model to manage requirement engineering for global software development. Kuwait Journal of Science, **46**: 33-42.

Amirfallah, A., Trautsch, F., Grabowski, J., & Herbold, S. (2019). A systematic mapping study of developer social network research. ArXiv, abs/1902.07499.

Avnet, M. S. (2016). A network based analysis of team coordination and shared cognition in systems engineering. Systems Engineering, **19**: 395-408.

Bernardi, M. L., Canfora, G., Lucca, G. A. D., Penta, M. D. & Distante, D. (2011). Do Developers Introduce Bugs When They Do Not Communicate? The Case of Eclipse and Mozilla. 2012 16th European Conference on Software Maintenance and Reengineering, 27-30 March 2012. 139-148.

Bettenburg, N. (2011). Mining development repositories to study the impact of collaboration on software systems. Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ACM, 376-379.

Betz, S., Mite, D., Fricker, S., Moss, A., Afzal, W., Svahnberg, M., Wohlin, C., Borstler, J. & Gorschek, T. (2013). An evolutionary perspective on socio-technical congruence: The rubber band effect. Replication in Empirical Software Engineering Research (RESER), 2013 3rd International Workshop on, IEEE, 15-24.

Bird, C., Murphy, B., Nagappan, N. & Zimmermann, T. (2011). Empirical software engineering at microsoft research. Proceedings of the ACM 2011 conference on Computer supported cooperative work, ACM, 143-150.

Bolici, F., Howison, J. & Crowston, K. (2009). Coordination without discussion? Socio-technical congruence and Stigmergy in Free and Open Source Software projects. Socio-Technical Congruence Workshop in conj Intl Conf on Software Engineering, Vancouver, Canada.

Brooks Jr, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2/E, Pearson Education India.

Calefato, F., Lanubile, F. & Novielli, N. (2017). A preliminary analysis on the effects of propensity to trust in distributed software development. Global Software Engineering (ICGSE), 2017 IEEE 12th International Conference on, IEEE, 56-60.

Cataldo, M. & Herbsleb, J. D. (2008). Communication networks in geographically distributed software development. Proceedings of the 2008 ACM conference on Computer supported cooperative work. San Diego, CA, USA: ACM.

Cataldo, M. & Herbsleb, J. D. (2008). Communication patterns in geographically distributed software development and engineers' contributions to the development effort. Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, ACM, 25-28.

Cataldo, M. & Herbsleb, J. D. (2013). Coordination breakdowns and their impact on development productivity and software failures. IEEE Transactions on Software Engineering, **39**: 343-360.

Cataldo, M., Herbsleb, J. D. & Carley, K. M. (2008). Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, ACM, 2-11.

Cataldo, M., Mockus, A., Roberts, J. A. & Herbsleb, J. D. (2009). Software dependencies, work dependencies, and their impact on failures. IEEE Transactions on Software Engineering, **35**: 864-878.

Cataldo, M., Wagstrom, P. A., Herbsleb, J. D. & Carley, K. M. (2006). Identification of coordination requirements: implications for the Design of collaboration and awareness tools. Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, ACM, 353-362.

Colfer, L. J. & Baldwin, C. Y. (2016). The mirroring hypothesis: theory, evidence, and exceptions. Industrial and Corporate Change, **25**: 709-738.

Conway, M. E. (1968). How do committees invent. Datamation, **14**: 28-31.

Datta, S. (2018). How does developer interaction relate to software quality? An examination of product development data. Empirical Software Engineering, **23**: 1153-1187.

De Santana, A. M., Da Silva, F. Q., De Miranda, R. C., Mascaro, A. A., Gouveia, T. B., Monteiro, C. V. & Santos, A. L. (2013). Relationships Between Communication Structure and Software Architecture: An Empirical Investigation of the Conway's Law at the Federal University of Pernambuco. Replication in Empirical Software Engineering Research (RESER), 2013 3rd International Workshop on, IEEE, 34-42.

Dingsøyr, T., Moe, N.B., Fægri, T.E. (2018). Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empir Software Eng **23**: 490–520 (2018). https://doi.org/10.1007/s10664-017-9524-2

Dingsøyr, T., Rolland, K., Moe, N. B., & Seim, E. A. (2017). Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development. Procedia Computer Science, **121**:123-128. doi:https://doi.org/10.1016/j.procs.2017.11.017

Ehrlich, K., Helander, M., Valetto, G., Davies, S. & Williams, C. (2008). An Analysis of Congruence Gaps and Their Effect on Distributed Software Development.

Fabbri, S., Silva, C., Hernandes, E., Octaviano, F., Di Thommazo, A. & Belgamo, A. (2016). Improvements in the StArt tool to better support the systematic review process. Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, ACM, **21**.

Gokpinar, B., Hopp, W. J. & Iravani, S. M. (2010). The impact of misalignment of organizational structure and product architecture on quality in complex product development. Management science, **56**: 468-484.

Golzadeh, M. (2019). Analysing socio-technical congruence in the package dependency network of Cargo. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Tallinn, Estonia: ACM.

Hameed, H., Khalid, H., Qamar, U., & Abass, S. K. (2017). Optimizing software project management staffing and work-force deployment processes using swarm intelligence, Computing Conference, London, 2017, pp. 78-84, doi: 10.1109/SAI.2017.8252084.

Ibrahim, Z., Md Johar, M. G., & Abdul Rahman, N. (2018). The Quality of Teamwork on Methodology in Software Development Workflow. International Journal of Engineering and Technology, Vol **7**: No 4.28 (2018), 510-525. doi: 10.14419/ijet.v7i4.28.22641

Jiang, L., Carley, K. M. & Eberlein, A. (2012). Assessing team performance from a socio-technical congruence perspective. Proceedings of the International Conference on Software and System Process, IEEE Press, 160-169.

Kiani, Z. U. R., Smite, D. & Riaz, A. (2013). Measuring Awareness in Cross-Team Collaborations-Distance Matters. Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on, IEEE, 71-79.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, **33**: 1-26.

Kitchenham, B. & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering.

Kuhrmann, M. & Münch, J. (2016). Distributed software development with one hand tied behind the back: a course unit to experience the role of communication in GSD. Global Software Engineering Workshops (ICGSEW), 2016 IEEE 11th International Conference on, IEEE, 25-30.

Kwan, I. & Damian, D. (2011). Extending socio-technical congruence with awareness relationships. Proceedings of the 4th international workshop on Social software engineering, ACM, 23-30.

Kwan, I., Schroter, A. & Damian, D. (2011). Does socio-technical congruence have an effect on software build success? A study of coordination in a software project. IEEE Transactions on Software Engineering, **37**: 307-324.

Kwan, I., Schröter, A. & Damian, D.(2009). A weighted congruence measure. Workshop on SocioTechnical Congruence, 1-4.

Li, T., Vedula, S. S., Hadar, N., Parkin, C., Lau, J., & Dickersin, K. (2015). Innovations in data collection, management, and archiving for systematic reviews. Ann Intern Med, **162**(4): 287-294. doi: 10.7326/m14-1603

Maheshwari, M., Kumar, U. & Kumar, V. (2012). Alignment between social and technical capability in software development teams: An empirical study. Team Performance Management: An International Journal, **18**: 7-26.

Marczak, S., Kwan, I. & Damian, D. (2009). Investigating collaboration driven by requirements in cross-functional software teams. Requirements: Communication, Understanding and Softskills, 2009 Collaboration and Intercultural Issues on, IEEE, 15-22.

Mcleod, L. & Macdonell, S. G. (2011). Factors that affect software systems development project outcomes: A survey of research. ACM Computing Surveys (CSUR), **43**: 24.

Nguyen, T., Wolf, T. & Damian, D. (2008). Global software development and delay: Does distance still matter? Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on, 2008. IEEE, 45-54.

Palyart, M., Murphy, G. C. & Masrani, V. (2018). A Study of Social Interactions in Open Source Component Use. IEEE Transactions on Software Engineering, **44**: 1132-1145.

Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M. & Beecham, S. (2014). Using agents to manage socio-technical congruence in a global software engineering project. Information Sciences, **264**: 230-259.

Portillo Rodríguez, J. (2013). An agent architecture to improve coordination and communication in GSE.

Sangwan, R. S. & Ros, J. (2008). Architecture leadership and management in globally distributed software development. Proceedings of the first international workshop on Leadership and management in software architecture, ACM, 17-22.

Sarma, A., Herbsleb, J. & Van Der Hoek, A. (2008). Challenges in measuring, understanding, and achieving social-technical congruence. Proceedings of Socio-Technical Congruence Workshop, In Conjunction With the International Conference on Software Engineering.

Sierra, J. M., Vizcaíno, A., Genero, M. & Piattini, M. (2018). A systematic mapping study about socio-technical congruence. Information and Software Technology, **94**: 111-129.

Šmite, D. & Galviņa, Z. (2012). Socio-technical congruence sabotaged by a hidden onshore outsourcing relationship: lessons learned from an empirical study. International Conference on Product Focused Software Process Improvement, Springer, 190-202.

Šmite, D., Moe, N. B., Šāblis, A. & Wohlin, C. (2017). Software teams and their knowledge networks in large-scale software development. Information and Software Technology, **86**: 71-86.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. Journal of Business Research, **104**: 333-339. doi:https://doi.org/10.1016/j.jbusres.2019.07.039

Sobri, W., Fauzi, S., Nasir, M., Ahmad, R. & Suali, A. (2017). A mechanism to assess the relationship between socio-technical congruence and project performance in incremental model. Journal of Fundamental and Applied Sciences, **9**: 58-74.

Sosa, M. E., Eppinger, S. D. & Rowles, C. M. (2004). The misalignment of product architecture and organizational structure in complex product development. Management science, **50**: 1674-1689.

Suali, A., Fauzi, S., Nasir, M. & Sobri, W. (2017). A brief method to assess the association of socio and technical dependencies on software quality. Journal of Fundamental and Applied Sciences, **9**: 101-114.

Syeed, M., Hammouda, I. & Berko, C.(2013). Exploring Socio-Technical Dependencies in Open Source Software Projects: Towards an Automated Data-driven Approach. Proceedings of International Conference on Making Sense of Converging Media, ACM, 273.

Tahir, M., Khan, F., Babar, M., Arif, F. & Khan, F. (2016). Framework for Better Reusability in Component Based Software Engineering. the Journal of Applied Environmental and Biological Sciences (JAEBS), **6**: 77-81.

Tamburri, D. A. A., Palomba, F. & Kazman, R. (2019). Exploring Community Smells in Open-Source: An Automated Approach. IEEE Transactions on Software Engineering, 1-1.

Valetto, G., Chulani, S. & Williams, C. (2008). Balancing the value and risk of socio-technical congruence. Workshop on Sociotechnical Congruence.

Valetto, G., Helander, M., Ehrlich, K., Chulani, S., Wegman, M. & Williams, C. (2007). Using software repositories to investigate socio-technical congruence in development projects. Proceedings of the Fourth International Workshop on Mining Software Repositories, IEEE Computer Society, **25**.

Van Eck, N. J. & Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. Scientometrics, **84**: 523-538.

Wagstrom, P., Herbsleb, J. D. & Carley, K. M.(2010). Communication, team performance, and the individual: bridging technical dependencies. Academy of Management Proceedings, Academy of Management Briarcliff Manor, NY 10510, 1-7.

Wang, X., Xiao, L., Yang, Y., Xu, X. & Jiang, Y. (2018). Identifying TraIn: a neglected form of socio-technical incongruence. Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings. Gothenburg, Sweden: ACM.

Wyman, O. (2003). The congruence model: a roadmap for understanding organizational performance. Oliver Wyman Group. http://www. oliverwyman.com/ow/pdf_files/ Congruence_Model_INS.

Yahyaoui, H., El-Qurna, J. & Almulla, M. (2020). Specification and recognition of service trust behaviors. Kuwait Journal of Science, **47**(1).

Zhang, F., Khomh, F., Zou, Y. & Hassan, A. E. (2012). An Empirical Study of the Effect of File Editing Patterns on Software Quality. 2012 19th Working Conference on Reverse Engineering, 15-18 Oct. 2012 2012. 456-465.

Zhang, W., Cheung, S. C., Chen, Z., Zhou, Y. & Luo, B. (2019). File-level socio-technical congruence and its relationship with bug proneness in OSS projects. Journal of Systems and Software, **156**: 21-40.

Zhang, W., Zhang, Q., Yu, B. & Zhao, L. (2014). Knowledge map of creativity research based on keywords network and co-word analysis, 1992–2011. Quality & Quantity, **49**.

Zhang, W., Chen, Z., & Luo, B. (2018). Does Socio-Technical Congruence Have an Effect on Continuous Integration Build Failures? An Empirical Study on 10 GitHub Projects. Paper presented at the 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS).

Submitted: 24/02/2020
Revised: 23/10/2020
Accepted: 16/11/2020
DOI: 10.48129/kjs.v49i1.9240