Systematic Literature Review: Causes of Rework in GSD

Shiza Nawaz, Anam Zai, Salma Imtiaz, and Humaira Ashraf
Department of Computer Science and Software Engineering, International Islamic University, Pakistan

Abstract: *Global Software Development (GSD) involves multiple sites which comprise of different cultures and time zones apart from geographical locations. It is a common software development approach adopted to achieve competitiveness. However, due to multiple challenges it can result in misunderstandings and rework. Rework raises the chance of project failure by delaying the project and increasing the estimated budget. The aim of this study is to identify and categorize the rework causes to reduce its frequency in GSD. To identify the empirical literature related to causes of rework, we performed a Systematic Literature Review (SLR). A total of 23 studies are included as a result of final inclusion. The empirical literature from the year 2009 to 2020 is searched. The overall identified causes of rework in GSD are categorized into 6 major categories which are communication, Requirement Management (RM), roles of stakeholders, product development/integration issues, documentation issues, and differences among stakeholders. The most reported rework causes are related to the category of communication & coordination and RM. Moreover, an industrial survey is conducted to validate the identified rework causes and their mitigation practices from practitioners. This study will help practitioners and researchers in addressing the identified causes and therefore reduce the chances of rework.*

Keywords: *Global software development, communication and coordination issues, requirement management issues and rework.*

Received August 21, 2020; accepted April 28, 2021
https://doi.org/10.34028/iajit/19/1/12

1. Introduction

Global Software Development (GSD) is a distributed development strategy that involves regions separated by space, time, and culture. It has become an industry norm due to the variety of advantages such as quality work, reduced development cost and around the clock development [18, 39]. The world has turned into a global village and globalization facilitates individuals from different areas to share ideas, thoughts, cultures, and business. The internet has opened new doors for business in the field of Information Technology (IT) and Engineering, where distance is of less importance [23]. Business globalization gives the concept of GSD which leads software development organizations to practice distributed development i.e., not restricted to a single country [4]. GSD ensures around the clock development, utilization of expertise from around the world, and therefore quality development but at the same time, it also faces many challenges as compared to co-located development [15].

Some of the challenges in GSD are due to the complexity of tasks, poor interpersonal relationship [25], difficulty of modules integration [13], and communication [3, 25, 37], poor planning of RM process, frequent changes in requirements [43], tool mismatch, outdated documentation [20], absence of product manager [12], different processes across distributed sites [15], etc., These challenges result in rework which delays the project and results in cost overrun.

"Rework" requires continuous attention and resolution throughout the Software Development Life Cycle (SDLC). Previous studies discuss different rework causes randomly. Our study aims to identify these causes from the empirical literature and categorize them. We suggest mitigation practices to minimize the root causes of rework.

In this study, Systematic Literature Review (SLR) is conducted to evaluate and investigate all available causes of rework in GSD from the year 2009 to 2020. A total of 27 causes are identified from 23 studies that are further classified into 6 major categories. Limited rework causes are discussed in previous literature therefore, an industrial survey is conducted to complement the data. The identified rework causes are categorized to promote comprehension for practitioners and researchers.

The organization of the paper is, section 2 is related work, section 3 describes the research methodology and section 4 gives detail of the SLR process. The findings of SLR are discussed in section 5 and results are explained in section 6. Threats to validity and industrial survey are discussed in sections 7, 8 respectively. Mitigation practices discussed in section 9. The research contribution is discussed in section 10. Finally, the conclusion and future work is in section 11.

2. Related Work

In Software Industry, rework is the most common cause of uncertainty and as a result, it is one of the most significant factors that negatively affect organization's ability to meet customer demands [8, 16]. It is a critical issue during the development of software [9, 17] because of extra effort of redoing a work that was initially poorly done [17, 38]. Literature highlights that fixing/correcting software flaws consume half of the effort and resources of the project [1]. According to multiple studies, redoing increases cost, generate delays and require extra time [16, 21] and directly impacts an organization's performance, productivity, and profit margins [38]. Most of the cost can approach or exceed 50 percent of the entire project cost [21]. The cost of rework rises significantly in case of increased delay, relative to the development life cycle i.e., between the incidence of an issue and its remedy [38].

Different practices are used in software industry to deal with rework. The usage of Capability Maturity Model (CMMI) and Agile practices when combined decreases rework and boosts productivity [8]. These solutions are effective in traditional software development but are not effective in case of GSD due to the spatial, temporal and cultural distance. The corporate structure also establishes boundaries and obstacles [2]. Rework is a common problem in GSD and requires more time, effort and cost due to the distributed nature of team members/teams. Researchers have identified different challenges [5, 6, 19, 22, 41] which are reported in literature.

The ineffective communication results in misinterpretation of user demands leading to the development of the incorrect features [42]. It can also lead to poor decision-making, and result in backlog management. It also results in technical faults, which could lead to defects and rework. These are just a few instances of how effective communication is critical and how poor communication leads to rework [42]. Different project management approaches might result in rework due to integration, and interoperability issues [2]. The design mistakes and omissions are the primary source of rework and contribute to 5.4% of the total construction cost [10]. Current development approaches result in unnecessary rework and cause mismatches between design and development [31]. The adequate and timely engagement of all stakeholders is critical during the negotiation of new requirements to minimize future rework effort [36]. All of these problems result in rework and disrupts the software development.

GSD is a trendy software development approach that many software organizations have adopted to save money and time. Rework, on the other hand, has a negative impact on these advantages. The rework is handled as a reactive approach i.e., strategies are used to perform rework while reducing the impact of rework. This reactive approach is not cost effective, since a lot of effort is wasted on handling the rework. Addressing rework in a proactive approach would require eliminating or minimizing the causes that generates rework. Doing so will reduce the likelihood of occurrence of rework and reduce the impact of rework as well saving much of the unnecessary effort. Rework is a problem that affects businesses of all sizes, hence a critical issue. To the best of our knowledge no empirical research on rework is published both in traditional software development and GSD. The work which reports the causes is limited and scattered. Since rework has severe and more negative consequences during distributed development, therefore we have chosen to identify, analyse and synthesize the causes of rework in GSD. The outcome of the research can be used by researchers as well as practitioners to ensure successful GSD.

3. Research Methodology

The research methods selected for this study are SLR and industrial survey. SLR is performed to identify relevant empirical data from the literature. It is complemented with an industrial survey to understand rework causes from practice. Under this survey, 25 requests are sent to organizations but only 18 organizations participated. The practitioners are also asked to report mitigation practices to address the rework causes. The Figure 1 below elaborates the research methodology.

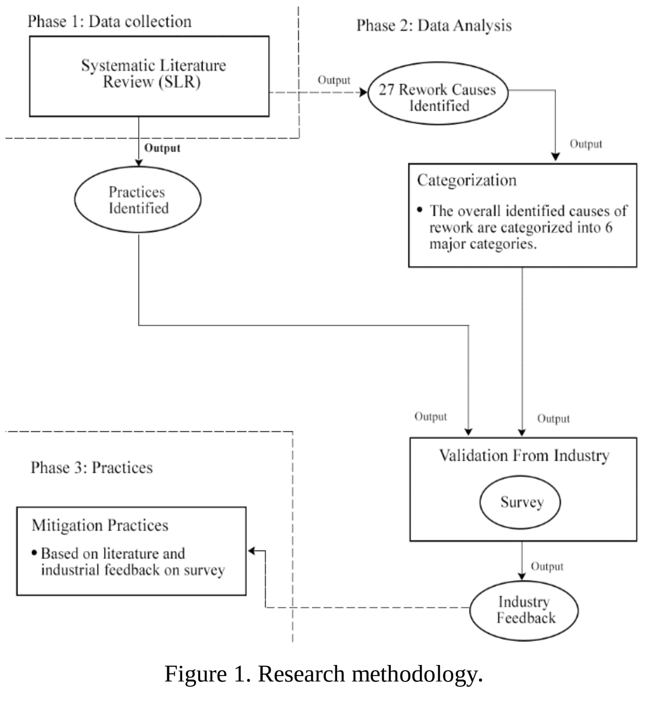

Figure 1. Research methodology.

3.1. Phase 1: Data Collection

Data for this research is collected by conducting SLR. In SLR we collected the rework causes which are faced during GSD. Practices are also found from selected studies to mitigate rework causes.

3.2. Phase 2: Data Analysis

Three major steps performed in this section:

1) *Categorization*: Six major causes are found after the categorization of the overall identified rework causes.
2) *Analysis of identified practices*: 2nd step is the analysis of mitigation practices that are found from literature.
3) *Validation of rework causes and practices from industry*: The last step of data analysis is to validate the rework causes and to get more mitigation practices from industry.

3.3. Phase 3: Mitigation Practices

The output of phase 2 is used to propose migration practices, which can help the practitioners to minimize the effects of identified causes.

4. Systematic Literature Review

SLR is the kind of secondary study [24] which provides a method of examining the empirical literature in any research field and the question of interest in a systematic manner [30]. SLR applies a systematic procedure of inclusion/exclusion, data extraction and, quality assessment. A protocol is designed prior to the execution of research, which is strictly followed, which in turn makes sure that the process is repeatable with the same results [29]. The steps that we followed to conduct the SLR are given in Figure 2.

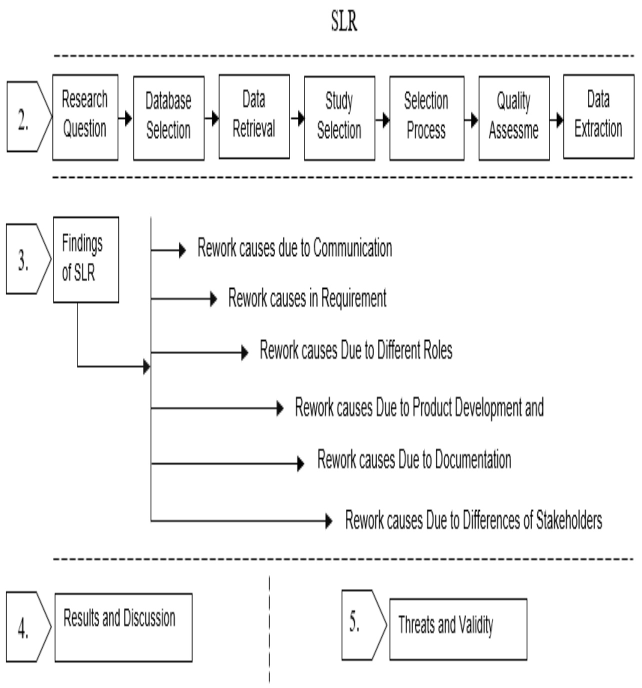
SLR
2.
Research Question
Database Selection
Data Retrieval
Study Selection
Selection Process
Quality Assessme
Data Extraction
3.
Findings of SLR
Rework causes due to Communication
Rework causes in Requirement
Rework causes Due to Different Roles
Rework causes Due to Product Development and
Rework causes Due to Documentation
Rework causes Due to Differences of Stakeholders
4.
Results and Discussion
5.
Threats and Validity

Figure 2. SLR process.

4.1. Research Question (RQ)

RQ is an initial step of SLR process and it should be defined on the basis of research objective. We have formulated the following RQ to find the causes of rework in GSD.

- RQ# 1: What are the causes of rework during Global Software Development?

4.2. Database Selection

We choose the following digital libraries to identify empirical literature because these are important databases related to computer science and software engineering:

- IEEE
- Science direct
- Springer link
- ACM digital library
- Others (scholar Google)

4.3. Data Retrieval

The data retrieval depends on the search string composed of the research question. In order to avoid researcher bias, the below given strategy is used to construct the main search string:

1. Analyse the research question and identify the main terms.
2. Identify synonyms for major terms (if exist).
3. Connect alternative synonyms with Boolean OR.
4. Link the main terms with Boolean AND.

The resultant main search string is:

((“GSD” OR “global software development” OR “DSD” OR “distributed software development”) AND (“rework” OR “redo” OR “modify” OR “reshape” OR “alter” OR “remodel” OR “revise” OR “remake” OR “rewrite” OR “change”) AND (“cause” OR “reason” OR “origin” OR “creator” OR “base” OR “source”))

This main term is customized for the different databases based on their format i.e., use of wildcards and brackets etc.

4.4. Study Selection

The selections of studies rely on the below given inclusion and exclusion criteria:

1) Included Studies should be

1. Published in journal or conference
2. Published between 2009-2020
3. Available in English
4. Regarding causes of rework in GSD

2) Excluded Studies should be

1. Duplicated or repeated study.
2. Not related to causes of rework in GSD.

3. Studies published in other languages.
4. Thesis and gray literature.

4.5. Selection Process

The Figure 3 shows the overall search process in detail from search string to the number of selected studies which are included at each level. The search process consists of two levels:

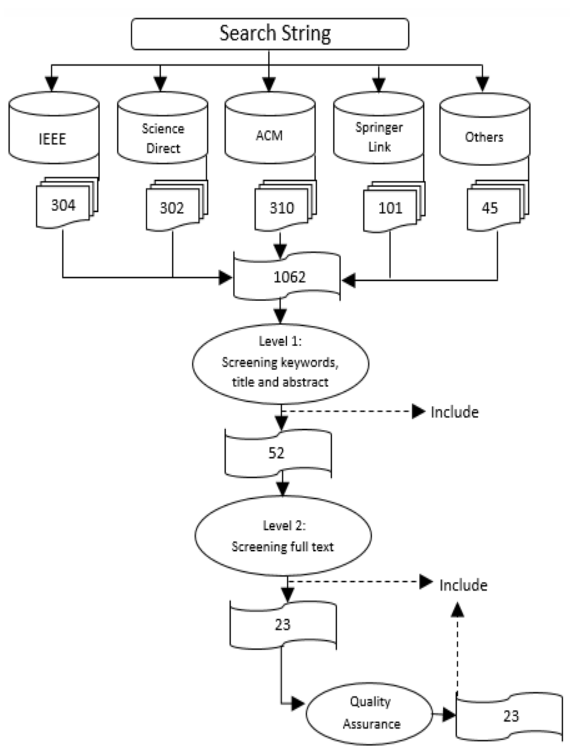

Figure 3. Detail of number of evidences at each step.

1) *Level* I: In level 1, the primary studies searched through selected sources by using the main search string defined in section 4.3 are included or excluded by reading titles and abstracts. The inclusion and exclusion criteria is used to make the final decision.
2) *Level* 2: In level 2 All the included papers collected in level 1 are studied in detail to identify the causes of rework in GSD or excluded based on the availability of relevant material or not.

4.6. Quality Assessment (QA)

According to [34], in SLR, QA is essential step, as it enhances reliability of selected studies in order to avoid research bias. QA criteria of selected studies is designed by using the guidelines defined by Kitchenhem and Charters is [24]. QA is based on following Quality Questions (QQ):

- QQ 1: Is the aim/objectives of the study clearly stated?
- QQ 2: Is the study designed to achieve defined objectives?
- QQ 3: Are findings and results clearly described in the study?
- QQ 4: Is the study well referenced?

The QA of an individual study is performed through quality points which are based on the following rules:

- *Rule* 1: If selected study answers the quality question then its score is 1.
- *Rule* 2: If a selected study partially answers the quality question then its score is 0.5.
- *Rule* 3: If a selected study does not answer the quality question then its score is 0.

The assessed quality of each of the selected study is given in Table 1 below. None of the study is excluded due to QA since the QA score is ok for all studies.

Table 1. Assessed the quality of each selected study.

PC	Q1	Q2	Q3	Q4	Total Points	Assessed Quality
PC1	1	1	1	1	4	Excellent
PC2	1	1	1	1	4	Excellent
PC3	1	1	0	1	3	Good
PC4	1	1	1	1	4	Excellent
PC5	1	1	1	1	4	Excellent
PC6	1	1	0.5	1	3.5	Excellent
PC7	0.5	0.5	0.5	1	2.5	Medium
PC8	1	0.5	0.5	1	3	Good
PC9	1	1	1	1	4	Excellent
PC10	1	1	1	1	4	Excellent
PC11	1	1	0.5	1	3.5	Excellent
PC12	1	1	1	1	4	Excellent
PC13	1	1	1	1	4	Excellent
PC14	1	1	1	1	4	Excellent
PC15	1	1	0.5	1	3.5	Excellent
PC16	1	1	1	1	4	Excellent
PC17	1	1	1	1	4	Excellent
PC18	1	1	0.5	1	3.5	Excellent
PC19	0.5	0.5	0.5	1	2.5	Good
PC20	1	1	0.5	1	3.5	Excellent
PC21	0.5	0.5	0.5	1	2.5	Good
PC22	1	1	1	1	4	Excellent
PC23	1	1	1	1	4	Excellent

Low: If total points <=1
Medium: If total points=>1&<=2
Good: If total points=>2&<=3
Excellent: If total points=>3&=4

4.7. Data Extraction

Microsoft Excel is used to create data extraction forms that recorded information on how the studies answered RQ. Metadata such as paper title, author, and publication's year, database, and the decision of inclusion or exclusion with rationale is recorded in data extraction forms. The number of studies included and excluded is detailed with help of Figure 3.

5. Findings of SLR

The data from SLR and industrial survey are analysed, and the causes of rework are reported, which are explained in detail one by one.

5.1. Rework Causes Due to Communication Issues

Communication is a major problem and it assumes complexity when teams are distributed globally. Table 2 presents the causes related to communication issues.

Table 2. Causes of rework due to communication issues.

Main causes	Sub-causes	PC	Frequency
Communication	Poor interpersonal relationship	PC3, PC5	2
	Task awareness	PC6, PC9, PC 23	3
	Availability awareness	PC18	1
	Phonetic spellings	PC14	1
	Lack of informalCommunication	PC1, PC 20	2
	Misinterpretation of vocabulary	PC4, PC13	2
	Poorly communicated requirements	PC9	1
	Poorly communicated module objectives	PC9	1

Poor interpersonal relationship between remote teams is one of the communication risks that increases the rework frequency [25, 27]. Communication barriers and awareness results in software integration and coordination breakdown that in turn leads to delay in task completion, and increased rework [37]. Task awareness at distributed location makes it challenging to find the right person. This in return leads to rework [22, 47]. Delay in response also increases rework [28]. When phonetic spellings are used by the client while providing requirements, it sometimes leads to rework [7]. The use of different terminologies leads to coordination time delay and rework [35]. Lack of Informal planning [45] and poorly communicated requirements or module objectives make it challenging to develop the correct modules, ultimately leading to rework [41, 47]. Misinterpretation of vocabulary due to cultural differences enhance chances of rework [20].

5.2. Rework Causes Due to Requirement Management Issues

Requirements Engineering (RE) is a very complex and human concerned activity; where the complexity increases in the case of GSD [28]. Table 3 presents the requirement management related causes.

Table 3. Causes of rework due to requirement management issues.

Main causes	Sub-causes	PC	Frequency
Requirement Management	Frequent change in requirements	PC8	1
	Requirements identified but not executed	PC8	1
	Poor planning of RMP	PC8	1
	Change un-notified to all stakeholders	PC16	1
	Mistakes at requirements elicitation phase	PC17, PC 19	2
	Problems in gaining a shared understanding	PC4, PC 21, PC 20	3
	Ineffective flow of changed information in RCM	PC2, PC15	2
	Tool mismatch	PC4	1

Developers at different sites face challenges in understanding the requirements [19, 33, 41]. Lack of effective collaborative tools and technologies makes RE more complicated [46]. Tool mismatch among team increases rework [20]. “RM is an organized activity to elicit, document and organize requirements of a software system, that maintain and establish a contract between the project development team and client on the changes to requirements” [43]. RM makes sure that all the collected requirements are implemented as well [43]. Shortcomings in this phase will result in unacceptable software and rework [40].

During SDLC, requirements constantly change [11] at every phase [27]. Its poor planning results in delay which in turn causes frequent changes and leads to rework [5]. Moreover, it impacts the design activity and increase the effort and cost [7]. Changes in requirements not notified to relevant stakeholders causes rework [32]. In RCM, the ineffective flow of changed information among distributed sites also increases rework chances [26].

5.3. Rework Causes Due to different Role

The software development literature highlights the importance of roles, to carry tasks related to the development and ensuring success. Table 4 presents the causes related to roles.

Table 4. Causes of rework due to different roles.

Main causes	Sub-causes	PC	Frequency
Different Roles	Stakeholder’s Unaligned agenda	PC11	1
	Absence of product manager	PC11	1

There are multiple shareholders involved in development of project who may have unaligned agendas, which results in delayed development and rework. In absence of product manager when nobody takes leadership and ownership of task from the group of stakeholders it also results in rework [12].

5.4. Rework Causes Due to Product Development and Integration Issues

In GSD development of product and integration of modules is a challenging task, however little attention is given to this phase in empirical literature. Table 5, shows identified causes of rework which are discussed in the empirical literature.

Table 5. Causes of rework due to product development and integration and management issues.

Main causes	Sub-causes	PC	Frequency
Product Development, Integration and management	Poor integration	PC7	1
	Mismatch of methodology and development process	PC4	1
	Mismatches in management approaches	PC13	1
	Project management challenges	PC 21	1
	Fast track projects	PC12	1

Module integration in dispersed environment is much problematic and most of the time results in rework [13]. According to [19], in GSD team also face challenges related to project management. Rework is a negative outcome, linked with poor integration strategy [20]. Mismatch of methodology and development process results in postponement [20]. The mismatches in the management approaches also result in rework [34]. In GSD fast track projects is a reason for overlapping dependent modules and thus, produce unexpected rework [14].

5.5. Rework Causes due to Documentation Issues

Unambiguous and updated documentation can be useful for resources therefore documentation should be verified. In system design, out-of-date documentation results in rework [20]. Table 6, discusses related issue from the empirical literature.

Table 6. Causes of rework due to documentation.

Main causes	Sub-causes	PC	Frequency
Documentation	Outdated documentation system design	PC4	1

5.6. Rework Causes due to differences of Stakeholders

The different national and organizational culture of distributed sites in GSD is a major source of rework [6, 41, 44]. The causes mentioned below in Table 7.

Table 7. Causes of rework due to differences of stakeholders.

Main causes	Sub-causes	PC	Frequency
Differences of stakeholders	National differences	PC10, PC 22	2
	Cultural differences	PC10,PC 20, PC 22	3
	Organizational differences	PC10, PC 22	2

6. Results and Discussion

This research reviews the existing empirical literature to identify reported GSD challenges which leads to rework. We selected SLR as our research method to overcome the biasness and to provide comprehensive overview of the available evidence. In the findings of SLR we identified different challenges. In order to identify challenges faced by the practitioners we conducted an industrial survey. Our proposed practices are based on the combined results of SLR and survey. In addition, the practices are validated from industry.

To address RQ1, the classification and categorization of scattered rework causes is performed and presented in Figure 4 below in the form of a fishbone diagram. The purpose is to pictorially represent the causes in meaningful categories for comprehension and further analysis.

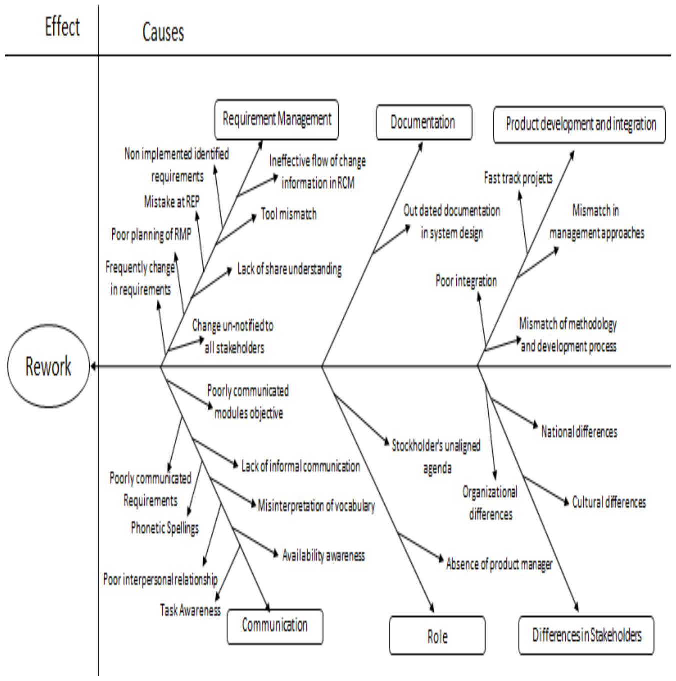
Effect
Causes
Requirement Management
Documentation
Product development and integration
Non implemented identified requirements
Ineffective flow of change information in RCM
Mistake at REP
Tool mismatch
Poor planning of RMP
Frequently change in requirements
Lack of share understanding
Change un-notified to all stakeholders
Out dated documentation in system design
Fast track projects
Mismatch in management approaches
Poor integration
Mismatch of methodology and development process
Rework
Poorly communicated modules objective
Lack of informal communication
Poorly communicated Requirements
Misinterpretation of vocabulary
Phonetic Spellings
Availability awareness
Poor interpersonal relationship
Task Awareness
Stockholder's unaligned agenda
Absence of product manager
National differences
Organizational differences
Cultural differences
Communication
Role
Differences in Stakeholders

Figure 4. Fish bone diagram (causes of rework organized in categories.

The above fish diagram (Figure 4) is a simple and clear representation of all identified causes of rework into different categories. In order to answer RQ1, we obtained 27 causes of rework from 23 selected studies from SLR. The ultimate aim of this research is to categories all identified causes; which are categorized into 6 major categories. We identified "communication" (30%), "requirement management" (30%), "product development and integration" (15%), "differences in stakeholders" (11%), "roles" (7%), documentation issues (3%), as major categories of rework cause in GSD. Most of the studies reported communication and RM as a significant reason for rework in GSD.

7. Threats to Validity

The threats to the study are due to access of limited resources for conducting SLR and also time constraint and limited budget, however we have tried to remove these limitations by

- Searching in related sources of computer sciences and software engineering.
- Use of synonyms to identify all empirical evidence related to rework in GSD.
- The search process is planned and piloted before execution.
- In addition, the organizations active in GSD are targeted to overcome threats.

8. Industrial Survey

To get practitioner opinions on the identified rework causes and mitigation practices, an industrial survey is performed. An industrial survey via questionnaire is selected because it is easy to collect the information from samples of the target audience. As the background of this study is GSD so, it is effective to use industrial survey method because in this way the selected sample can be contacted easily via email, skype or telephone.

8.1. Population

The target audience of the survey are practitioners, who are working in GSD based organizations.

8.2. Sampling

The type of sampling chosen is non-probability sampling and snowballing technique for data collection. We do not have a predefined list of distributed organizations due to the vast population of GSD organizations as well as inability to reach all in person.

8.3. Instrument

A survey is conducted by using questionnaires as an instrument for validation which is based on two steps. In the first step, validation of the overall rework causes performed via structured (closed-ended) questions. In second step, open-ended and partially structured questions are used to validate the identified practices and to get some more practices as feedback to mitigate rework issues in GSD.

8.4. Survey Instrument Administration

Survey administrated via email, social media (skype) and direct meeting (unstructured interviews) to personal contacts.

8.5. Survey Result

The main purpose behind conducting the industrial survey is to get practitioners' feedback on the causes of rework in GSD. Additionally, in the result section of the survey some mitigation practices to reduce rework causes are found. A detailed discussion of these practices is given in section 9.

8.6. Limitation of Survey

Most of the survey is based on personal contacts and snowballing strategy, where the first link is asked for more referrals. Therefore the survey is limited due to number of responses received.

9. Proposed Practices to Reduce Rework Causes

There are number of practices found from literature and industrial survey to address the identified causes of rework. Some of the practices are discussed below:

- **P1-Direct communication**:

In GSD due to lack of group cohesiveness, the communication between teams present at distributed sites becomes weak and causes communication challenges, which can be resolved through continuous feedback. Therefore, initial level of software development direct communication is required to build strong relationship among the team members.

- **P2-Use Administrative tools and agile development process**:

By personal awareness means that we are aware of who is working on the specific task and in which timings that person is available. The rework due to task unawareness can be minimized by using common knowledge, repositories and administrative tools (JIRA etc.,). These tools give central visualization of all scheduled activities with details of all resources. The agile development process is also helpful for task awareness.

- **P3-Established communication plan**:

To ensure availability awareness and to avoid communication bottlenecks, communication process must be planned according to situation and continuously examined after implementation.

- **P4-Discussions/Meetings**:

Quality assurance procedures and plans should be discussed early in the GSD process and clear module objectives are communicated before implementations to get feedback in the early stages.

- **P5-Review Formal Document**:

Formal documents should be reviewed by content writer or language expert to avoid issues due to phonetic spellings.

- **P6-Language training lessons**:

Misinterpretation of vocabulary can be reduced by choosing sites in culturally similar locations and also provide language training lessons to reduce linguistic distance.

- **P7-Extensive use of synchronize communication technology**:

Informal communication is a challenge in GSD environment, but it can be minimized by using synchronous communication technology extensively.

- **P8- Promote visits and exchanges among sites**:

Exchange team members among sites and also promote visits to overcome the lack of informal communication.

- **P9- Management of changed information**:

Follow standard approaches in RCM and manage changes in requirements at the right time to overcome ineffective flow of change information.

- **P10- Change notify to all stakeholders**:

Change in requirements should timely be propagated and the status be clearly informed to the relevant stakeholders.

- **P11- Implement RM process**:

Sometimes identified requirements are not executed due to poor RM process, therefore use a standard RM process to ensure and validate implementation of all requirements.

- **P12- Requirement verification**:

Requirement (documentation) verification also helps when requirements are identified but not executed. Therefore, all requirements should be noted down and verified from the client.

- **P13- Use incremental working solution**:

Change in requirement is obvious in the GSD environment and we cannot resist or stop it. At the initial level of software, development change is good but frequent late changes lead to rework. Use prototype or incremental working solutions to overcome problems due to frequent late changes in requirements. Different online applications for real scenarios are also helpful for better understandings of prototypes.

- **P14- Established business analyst team**:

To improve the RM process the established business analyst team arranges direct project planning meetings before implementation. Metrics are useful for the effectiveness of RMP.

- **P15- Direct communication between users and developers**:

Extensive use of synchronous communication technology or face to face communication among users and developers is important to get the correct requirements initially and minimize mistakes at the requirement elicitation phase. A well-defined process (agile) and the tool can be used to acquire requirements.

- **P16- Continuous discussion**:

Continuous discussion of relevant stakeholders is required for the development of mutual understanding.

- **P17- Develop core team**:

Develop a core team for each product and schedule group meetings for the common understanding of scope, immediately after getting requirements.

- **P18- Follow defect prevention activities**:

Defect prevention activities help to improve module integration.

- **P-19- Plan fast track projects**

Usually, fast track techniques used for small projects but there is a chance of rework in case of dependency between modules. For this proper planning and alternate solutions required.

- **P-20- Improve code review process**

Outdated documentation is an issue which requires document verification with code in the code review process.

10. Research Contribution

This research makes four primary contributions.

- First contribution is the design and execution of an empirical study to identify and categorize rework causes which are described in the section 5 (findings of SLR).
- The second contribution is the validation of the identified causes from the real world via survey (questionnaire and unstructured interviews).
- The third contribution is the proposed practices based on the combined results of SLR and survey. These practices will help to reduce rework causes described in section 9.
- The fourth contribution is the validation of proposed practices from practitioners.

11. Conclusions and Future Work

In this study, SLR is performed to identify the rework causes faced by distributed teams during the Software Development Lifecycle. The objective of the study is to classify and categorize the identified causes, which is the main contribution of this research. A total of 6 major rework categories and 27 causes are identified from the empirical literature.

The causes are categorized into main categories as part of this research work, which will also help in comprehension and evaluation. These causes may have a relationship with each other and impact each other. A detailed analysis will be performed in the future to identify the correlation among these causes.

The most reported rework causes are related to the category of communication and RM. The outcomes achieved from this SLR and industrial survey can be useful for practitioners and researchers to work on mitigating the causes of rework.

References

[1] Adnan F. and Andnaqvi I., "Effect of Rework on Project Success," *Science International*, vol. 27, no. 1, pp. 575-580, 2015.

[2] Ahmed M., Ateeq A., Khan S., and Junad M., "Software Cost Estimation in Global Software Development Business: A Review of Models and

Cost Drivers for Economical Business," *International Journal of Business and Economic Affairs*, vol. 6, no. 3, pp. 118-129, 2021.

[3] Al-Zaidi A. and Qureshi R., "Global Software Development Geographical Distance Communication Challenges," *The International Arab Journal of Information Technology*, vol. 14, no. 2, pp. 215-222, 2017.

[4] Aranda N., Vizcaino A., and Piattini M., "A Framework to Improve Communication During The Requirements Elicitation Process in GSD Projects," *Requirements Engineering*, vol. 15, vol. 4, pp. 397-417, 2010.

[5] Barbosa O., Albuquerque W., Bandeira A., Pivoto U., Pires F., and Bonifacio B., "Developing a Release Management Tool to Support Global Software Development: An Experience Report on Android Platform," *in Proceedings of the 15th International Conference on Global Software Engineering*, Seoul Republic of Korea, pp. 117-121, 2020.

[6] Beecham S., Clear T., Lal R., and Noll J., "Do Scaling Agile Frameworks Address Global Software Development Risks? an Empirical Study," *Journal of Systems and Software*, vol. 171, pp. 110823, 2021.

[7] Beecham S., Noll J., and Richardson I., "Using Agile Practices to Solve Global Software Development Problems-A Case Study," *in Proceedings of the IEEE International Conference on Global Software Engineeering Workshops*, Shanghai, pp. 5-10, 2014.

[8] Canedo D. and Santos A., "Factors Affecting Software Development Productivity: An Empirical Study," *in Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pp. 307-316, 2019.

[9] Cass A., Sutton S., and Osterweil L., "Formalizing Rework in Software Processes," *in Proceedings of the 9th International European Workshop on Software Process Technology*, Helsinki, pp. 16-31, 2003.

[10] De Beer J. and Depew C., "The Role of Process Engineering in the Digital Transformation," *Computers and Chemical Engineering*, vol. 154, pp. 107423, 2021.

[11] De Farias junior I., De Azevedo R., De Moura H., and Da Silva D., "Elicitation of Communication Inherent Risks in Distributed Software Development, *in Proceedings of the EEE 7th International Conference on Global Software Engineering Workshops*, Porto Alegre, pp. 37-42, 2012.

[12] Ebert C. and Brinkkemper S., "Software Product Management-An Industry Evaluation," *Journal of Systems and Software*, vol. 95, pp. 10-18, 2014.

[13] Faizan M., Ulhaq S., and Khan A., "Defect Prevention and Process Improvement Methodology for Outsourced Software Projects," *Middle-East Journal of Scientific Research*, vol. 19, no. 5, pp. 674-682, 2014.

[14] Fruchter R., Bosch-Sijtsema P., and Ruohomäki V., "Tension between Perceived Collocation and Actual Geographic Distribution in Project Teams," *Ai and Society*, vol. 25, no. 2, pp. 183-192, 2010.

[15] Galviņa Z. and Šmite D., "Software Development Processes in Globally Distributed Environment," *Scientific Papers, University of Latvia*, vol. 770, pp. 7-14, 2011.

[16] Giuseppe S. and Imanol U., "Do We Rework? A Path to Manage One of the Primary Cause of Uncertainty in Software Industry," *in Proceedings of International Conference on the Quality of Information and Communications Technology*, Ciudad Real, pp. 179-192, 2019.

[17] Hossain S., "Rework and Reuse Effects in Software Economy," *Global Journal of Computer Science and Technology*, vol. 18, no. 4, pp. 35-50, 2018.

[18] Imtiaz S. and Ikram N., "Dynamics of Task Allocation in Global Software Development," *Journal of Software: Evolution and Process*,vol. 29, no. 1, pp. e1832, 2017.

[19] Jain R. and Suman U., "A Project Management Framework for Global Software Development," *ACM SIGSOFT Software Engineering Notes*, vol. 43, no. 1, pp. 1-10, 2018.

[20] Jain R. and Suman U., "A Systematic Literature Review on Global Software Development Life Cycle," *ACM Sigsoft Software Engineering Notes*, vol. 40, no. 2, pp. 1-14, 2015.

[21] Jayatilleke S. and Lai R., "A Method of Assessing Rework for Implementing Software Requirements Changes," *Computer Science and Information Systems*, vol. 18, no. 1, pp. 32-32, 2020.

[22] Jiménez M., Piattini M., and Vizcaíno A., "Challenges and Improvements in Distributed Software Development: A Systematic Review," *Advances in Software Engineering*, vol. 2009, 2009.

[23] Kaur A., *Outsourcing Software Quality*, Institutt for Datateknikk og Informasjonsvitenskap, 2013.

[24] Kitchenham B. and Charters S., Guidelines for performing systematic Literature Reviews in Software Engineering," Technical Report, Keele University, 2007.

[25] Khan A., Basri S., and Selvam D., "Communication Risks in Gsd During Rcm:

Results fsrom Slr," *in Proceedings of the International Conference on Computer and Information Science*s, Kuala Lumpur, pp. 1-6, 2014.

[26] Khan A., Basri S., and Dominic D., "A Propose Framework Fsor Requirement Change Management In Global Software Development," *in Proceedings of International Conference on Computer and Information Science*, Kuala Lumpur, pp. 944-947, 2012.

[27] Khan B. and Ahsan A., "Recommended Configuration Management Practices for Freelance Software Developers," *in Proceedings of 5th International Conference on Software Engineering and Service Science*, Beijing, pp. 111-115, 2014.

[28] Khan H., Ahmed A., and Johansson C., "Requirements Understanding in Global Software Engineering: Industrial Surveys," *in Proceedings of International Conference on Computer and Software Modelling*, Singapore, 2011.

[29] Kitchenham B., "Procedures for Performing Systematic Reviews," Technical Report, Keele University, 2004.

[30] Kitchenham B., Brereton O., Budgen D.,Turner M., Bailey J., and Linkman S., "Systematic Literature Reviews in Software Engineering-A Systematic Literature Review," *Information and Software Technology*, vol. 51, no. 1, pp. 7-15, 2009.

[31] Leiva G., Maudet N., Mackay W., and Beaudouin-Lafon M., "Enact: Reducing Designer-Developer Breakdowns When Prototyping Custom Interactions," *ACM Transactions on Computer-Human Interaction*, vol. 26, no. 3, pp. 1-48, 2019.

[32] Minhas N. and Zulfiqar A., "An improved Framework for Requirement Change Management in Global Software Development," *Journal of Software Engineering and Applications*, vol. 7, no. 9, pp. 779-790, 2014.

[33] Nakatsu R. and Iacovou C., "A Comparative Study of Important Risk Factors Involved in Offshore and Domestic Outsourcing of Software Development Projects: A Two-Panel Delphi Study," *Information and Management*, vol. 46, vol. 1, pp. 57-68, 2009.

[34] Nguyen-Duc A., Cruzes D., and Conradi R., "The Impact of Global Dispersion on Coordination, Team Performance and Software Quality-A Systematic Literature Review," *Information and Software Technology*, vol. 57, pp. 277-294, 2015.

[35] Petersen K., Feldt R., Mujtaba S., and Mattsson M., "Systematic Mapping Study in Software Engineering," *in Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Italy, pp. 68-77, 2008.

[36] Qureshi S., Khan S., Iqbal J., and Ur-Rehman I., "A Study on Mitigating the Communication and Coordination Challenges During Requirements Change Management in Global Software Development," *IEEE Access*, vol. 9, pp. 88217-88242, 2021.

[37] Ramasubbu N., Cataldo M., Balan R., and Herbsleb J., "Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits," *in Proceedings of the 33rd International Conference on Software Engineering*, Honolulu, pp. 261-270. 2011.

[38] Ramdoo V. and Huzooree G., "Strategies to Reduce Rework in Software Development on an Organisation in Mauritius," *International Journal of Software Engineering and Applications*, vol. 6, no. 5, pp. 9-20, 2015.

[39] Ramos J., Esteban A., García J., and Amescua A., "Skills and Abilities for Working in A Global Software Development Team: A Competence Model," *Journal of Software: Evolution and Process*, vol. 26, no. 3, pp. 329-338, 2014.

[40] Sabahat N., Iqbal F., Azam F., and Javed M., "An Iterative Approach for Global Requirements Elicitation: A Case Study Analysis," *in Proceedings of International Conference on Electronics and Information Engineering*, Kyoto, pp. 361-366, 2010.

[41] Saleem N., Mathrani S., and Taskin N., "Investigating Critical Success Factors of Project Management in Global Software Development: A Work in Progress," *in Proceedings of in International Conference on Information Resources Management*, Auckland, 2019.

[42] Sedano T., Ralph P., and Péraire C., *Rethinking Productivity in Software Engineering*, Springer, 2019.

[43] Venkatesh B. and Balani L., "Requirement Management A Key To Successful Project Management for Software Systems," *An International Refereed Journal for Change and Development*, vol. 05, no.1, pp. 49-51, 2016.

[44] Verner J., Breretona O., Kitchenhama B., Turnera M., and Niaziacd M., "Risks and Risk Mitigation in Global Software Development: A Tertiary Study," *Information and Software Technology*, vol. 56, no. 1, pp. 54-78, 2014.

[45] Yadav V., "A Flexible Management Approach for Globally Distributed Software Projects," *Global Journal of Flexible Systems Management*, vol. 17, no. 1, pp. 29-40, 2016.

[46] Yaseen M., Baseer S., and Sherin S., "Critical Challenges for Requirement Implementation in Context of Global Software Development: A Systematic Literature Review," *in Proceedings of the International Conference on Open Source Systems and Technologies*, Lahore, pp. 120-125, 2015.

[47] Zafar A., Ali S., and Shahdzad R., "Investigating Integration Challenges and Solutions in Global Software Development," *in Proceedings of Frontiers of Information Technology*, Islamabad, pp. 291-297, 2011.

Appendices

This report contains a review, collected from defined data sources. This method is implemented by using the above (section 4.4) defined inclusion and exclusion criteria.

Table 8. Paper types.

Paper type	Type Code
Article	A
Book	B
International Conference	IC
Journal	J
Thesis	T

Table 9. Detail of final selected studies.

PC	Type	Database	Publication year	Address
PC1	J	Springer	2015	Y
PC2	J	Science Direct	2015	Y
PC3	IC	Science Direct	2013	Y
PC4	A	ACM	2015	Y
PC5	IC	IEEE	2014	Y
PC6	IC	ACM	2011	Y
PC7	J	Other	2014	Y
PC8	J	Other	2016	Y
PC9	J	IEEE	2011	Y
PC10	J	Science Direct	2013	Y
PC11	J	Science Direct	2014	Y
PC12	J	Springer	2009	Y
PC13	J	Science Direct	2014	Y
PC14	IC	IEEE	2014	Y
PC15	IC	IEEE	2012	Y
PC16	A	Other	2015	Y
PC17	IC	IEEE	2011	Y
PC18	IC	Other	2010	Y
PC 19	IC	Other	2020	Y
PC 20	IC	Other	2019	Y
PC 21	J	ACM	2018	Y
PC 22	J	Other	2021	Y
PC 23	A	Other	2009	Y

Table 10. Paper codes.

PC	Paper Title
PC1.	"A Flexible Management Approach for Globally Distributed Software Projects."
PC2.	"A method of requirements change management for global software development."
PC3.	"A Proposed Framework for Communication Risks during RCM in GSD."
PC4.	"A Systematic Literature Review on Global Software Development Life Cycle."
PC5.	"Communication Risks in GSD during RCM: Results from SLR."
PC6.	"Analysis of Project Productivity, Quality, and Profits."
PC7.	"Defect Prevention and Process Improvement Methodology for Outsourced Software Projects."
PC8.	"Requirement management a key to successful project management for software systems."
PC9.	"Investigating Integration Challenges and Solutions in Global Software Development."
PC10.	"Risks and risk mitigation in global software development: A tertiary study."
PC11.	"Software product management – An industry evaluation."
PC12.	"Tension between perceived collocation and actual geographic distribution in project teams."
PC13.	"The impact of global dispersion on coordination, team performance and software quality – A systematic literature review."
PC14.	"Using Agile practices to solve Global Software Development problems – A Case Study."
PC15.	"A Propose Framework for Requirement Change Management in Global Software Development."
PC16.	"An Improved Framework for Requirement Change Management in Global Software Development."
PC17.	"An Iterative Approach for Global Requirements Elicitation: A Case Study Analysis."
PC18.	"Requirements Understanding in Global Software Engineering: Industrial Surveys."
PC 19	"Developing a Release Management Tool to Support Global Software Development."
PC 20	"Investigating Critical Success Factors of Project Management in Global Software Development: A Work in Progress."
PC 21	"A Project Management Framework for Global Software Development."
PC 22	"Do scaling agile frameworks address global software development risks? An empirical study."
PC 23	"Challenges and Improvements in Distributed Software Development: A Systematic Review

- **Analysis of selected studies**: Table 11 shows the codes of causes and sub-causes of rework which are used in Table 12 for analysis. By using codes all sub-causes are marked in a particular paper to provide us the overall view.

Table 11. Causes code.

Main Cause	**Main Cause-code**	**Sub Cause**	**Sub Cause-code**
Communication cause	CC	Poor interpersonal relationship	CC1
		Task Awareness	CC2
		Availability Awareness	CC3
		Phonetic spellings	CC4
		Lack of informal communication	CC5
		Misinterpretation of technical vocabulary	CC6
		Poorly communicated requirements	CC7
		Poorly communicated module objectives	CC8
Requirement Management	RMC	Frequent change in requirements	RMC 1
		Non implementation of identified requirements	RMC2
		Poor planning of RMP	RMC3
		Change un-notified	RMC4
		Mistakes at requirements elicitation phase	RMC5
		Lack of shared understanding	RMC6
		Ineffective flow of changed information in RCM	RMC7
		Tool mismatch	RMC8
Roles	RC	Stakeholder's Unaligned agenda	RC1
		Absence of product manager	RC2
Product Development and Integration	PDIC	Poor integration	PDIC1
		Mismatch of methodology and development process	PDIC2
		Mismatches in management approaches	PDIC3
		Project management challenges	PDIC4
		Fast track projects	PDIC5
Documentation	DC	Outdated documentation in system design	DC1
Differences of Stakeholders	DSC	National differences	DSC1
		Cultural differences	DSC2
		Organizational differences	DSC3

Table 12. Paper analysis.

PC / Sub cause-code	**PC 1**	**PC 2**	**PC 3**	**PC 4**	**PC 5**	**PC 6**	**PC 7**	**PC 8**	**PC 9**	**PC 10**	**PC 11**	**PC 12**	**PC 13**	**PC 14**	**PC 15**	**PC 16**	**PC 17**	**PC 18**	**PC 19**	**PC 20**	**PC 21**	**PC 22**	**PC 23**
CC1			✓		✓																		
CC2						✓			✓														✓
CC3																		✓					
CC4														✓									
CC5	✓																			✓			
CC6				✓									✓										
CC7									✓														
CC8									✓														
RMC1								✓															
RMC2								✓															
RMC3								✓															
RMC4																✓							
RMC5																	✓		✓		✓		
RMC6				✓																✓			
RMC7		✓													✓								
RMC8				✓																			
RC1											✓												
RC2											✓												
PDIC1							✓																
PDIC2				✓																			
PDIC3													✓										
PDIC4													✓										
PDIC5											✓												
DC1				✓																			
DSC1										✓												✓	
DSC2										✓										✓		✓	
DSC3										✓												✓	

The Table 12 below represents the analysis of each selected study, columns represent the paper-code of each study and rows of entire table represent identified causes.

Shiza Nawaz is a PhD student at IIU Islamabad, Pakistan. She holds a Bachelor degree of software engineering from University of AJK Muzaffarabad in 2014, and a Master degree in Software Engineering from IIUI in 2019. As academician, here research interests are: Global Software Development, Requirement Engineering, and Block chain. She is teaching as lecturer at IIUI.

Anam Zai holds her Bachelor's degree of Computer Science from FUUAST, Islamabad Pakistan in 2016 and a Master's degree in Software Engineering from IIU Islamabad, Pakistan in 2019. She is pursuing her Ph.D. in Software Engineering from the same University and her doctoral work involves Blockchain Technology, Information Security, GSD and Agile. She is a Lecturer at IIUI and, also working as "Specialist Information Security Risk" in banking sector.

Salma Imtiaz holds a PhD (Computing) Degree from Riphah International University, Islamabad. She is an Assistant Professor at the Department of Computer Science and Software Engineering (DCSSE), International Islamic University, Islamabad. She works in the area of Global Software Development, Software Requirement Engineering, Empirical Software Engineering and Agile Software Development. She heads the Software Engineering Research Group at International Islamic University. She has presented at national and international conferences and serves as a reviewer to many conferences and journals.

Humaira Ashraf received BCS degree from Balochist an University, Quetta, Pakistan in 2005 and MSCS degree from BUITEMS, Quetta, Pakistan in 2008. She received the Ph.D. degree in computer science (with majors in cellular mobile networks and wireless networks) from International Islamic University Islamabad, Pakistan in 2017. She has published several papers in Impact Factor Journals and International Conferences. She is also reviewer of many ISI-indexed and Impact Factor Journals. Her areas of interest include Wireless Sensor Networks, Next Generation Networks, Internet of Things, Network Security, IP Multimedia Sub-system, Voice over LTE and Voice over IP