Exploring the Aspects of Rework in Global Software Development

Ritu Jain
Assistant Professor, Medi-Caps University,
Indore, MP, India

Ugrasen Suman
Professor, SCSIT, Devi Ahilya University,
Indore, MP, India

ABSTRACT
Global software development (GSD) is a software development setting in which practitioners residing in different countries work together as a team to accomplish a software project. This paradigm is rapidly adopted by numerous software companies in order to reduce cost and time. However, these cost savings are seldom achieved due to geographical, temporal, socio-cultural and linguistic distances. These distances often cause misinterpretations and conflicting perceptions about the product to be built and often induce lots of rework. Excessive rework significantly increases risk of project failure. However, little research has been conducted on the aspects of rework in GSD. In order to reduce rework in GSD, aspects of rework need to be conceptualized. Also, cost associated with rework need to be measured. Thus, in this paper rework cycle and its associated drivers are proposed for GSD setting. A metric is also proposed to calculate cost of rework in GSD. The research work has also been validated through industrial survey.

Keywords
Global software development, rework cycle, rework cost, rework drivers, rework metric.

1. INTRODUCTION
GSD is a prevalent software development trend adapted by numerous software companies to accrue cost and time savings. Overall cost of a software project can be reduced by exploiting salary differences among members belonging to different economies. This approach can also reduce overall development time by facilitating round the clock development due to differences in time zone among different locations. However, these benefits are partially achieved due to geographical, temporal, socio-cultural and organizational distances. Geographical distance among team members inhibits formal as well as informal communication, project awareness, effective coordination, trust, and knowledge management [1]. Temporal distance among development sites increases feedback time, decreases synchronous communication, limit coordination and collaboration [2].

Team members belonging to different countries have different socio-cultural values, varied frame of references, incompatible work ethics which can lead to in-cohesiveness and distrust. It also limits communication and increases the probability of misunderstandings due to differences in native language and accent. Sometimes, team members also belong to different organizations and thus can face issues related to mismatch of processes, tools, and work culture, which can lead to organizational distance. The discrepancies due to geographical, temporal, socio-cultural, and organizational distances often lead to ambiguities, conflicts, defects and subsequently lots of rework during software development [1]. Approximately 40-50% of the software development effort is being wasted on rework [5, 6, 7]. This rework can eventually lead to failure of projects [3, 4].

Rework can be generated due to substandard software development process, ineffective project management techniques, and loose practitioner attitude [8, 9, 10]. Effectiveness of a software development process can be immensely improved by reducing the rework [5]. According to lean principles, rework is considered as one of the seven wastes which is experienced during software development. Boehm et al has identified avoidance of rework as one of the main strategies to improve productivity, reduce development time, and cost [13]. It was also found that among several approaches for reducing software development effort, initiatives for reducing rework provide maximum returns [5]. Still, GSD project management neither considers rework during planning nor measures it during project execution. Thus, it needs to be reduced in order to improve efficiency and productivity [11, 12]. In order to reduce rework during software development, it is necessary to understand the concept of rework with regard to global software development. Thus, we have proposed a rework cycle for GSD to conceptualize the concept of rework for GSD projects. We have also identified rework drivers which can influence the amount of rework in GSD setting. A rework cost metric which can be used to calculate the amount of effort expended for rework in a module is formulated for GSD projects. Finally, the proposed concepts are validated through industrial survey. Thus, this research would set a ground for researchers to further investigate the aspects of rework in GSD. It can help practitioners to understand the aspect of rework in GSD in order to save valuable cost, time, and effort of GSD projects

The paper is organized as follows. Section 2 presents a brief overview of rework in software development and summarizes the related work. Section 3 presents the proposed rework cycle and associated rework drivers for GSD. Section 4 describes the proposed metric for calculating cost of rework. Section 5 presents industrial validation of the proposed concepts and finally, section 6 provides concluding remarks.

2. BACKGROUND AND RELATED WORK
Rework implies to an activity of redoing or modifying a work which was implemented previously. In software development setting, rework to some extent is inevitable. However, excessive rework (more than 20%) could indicate problems in software development process and project management activities whereas, meagre rework (less than 10%) could be a sign of inadequate reviews, inspections, testing, and refactoring [7].

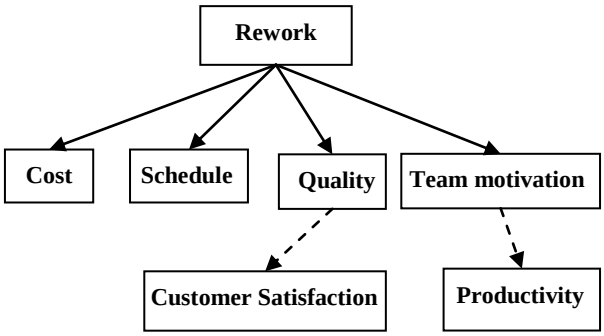

Figure 1 Negative Impact of Rework in Software Development

A project is considered as successful, if it is completed according to the planned schedule as well as budget and satisfies customers. However, excessive rework results into cost and schedule overrun. Recurrent rework deteriorates quality and thus, impedes customer satisfaction. It also adversely affects team morale and eventually reduces its productivity [7]. Thus, rework negatively affects the succeeding factors of a project. Figure 1 illustrates the negative impact of rework in software development setting.

Scant research has been performed on aspects of rework in software engineering whereas, in GSD, aspects of rework is still unexplored. Several researchers have investigated rework for traditional software development setting. Tonnellier et al. illustrated a model for rework, quantified it with the help of mathematical formulation, and suggested a metric for Thales Airborne Systems [14]. Basili et al. characterized and modeled the cost of rework for a library of reusable components. The model proposed by them predicts the rework cost associated with the component version and classifies it into either low rework cost or high rework cost category [15].

Damm et al. presented a case study-based model, which analyzes anomaly metrics to reduce rework in software development projects. This model detects group of anomalies that could have been prevented or fixed less costly. It identifies improvement areas for rework reduction [16]. However, none of the research in software engineering domain has explicitly investigated the concept of rework cycle and cost associated with it for GSD projects. Thus, this paper attempts to explore these aspects of rework in GSD. It would draw attention of researchers as well as practitioners to further investigate this software waste to improve the effectiveness of GSD process.

3. THE REWORK CYCLE

Even though rework is a recurrent software development activity in GSD, existing project planning and monitoring methods neither acknowledge nor measure it. In order to articulate the concept of rework in GSD, we have proposed the concept of vicious rework cycle during flow of work in a global software project.

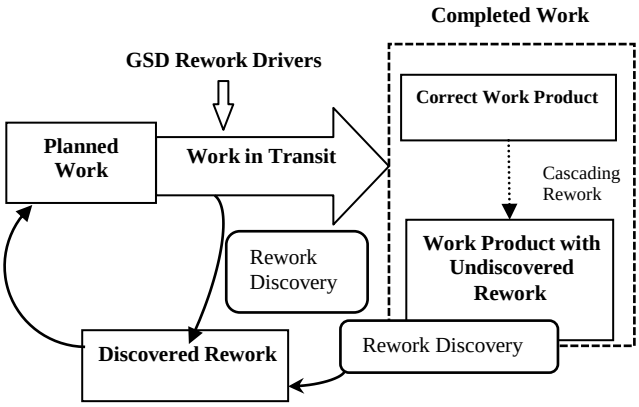

Figure 2 The Rework Cycle in GSD projects

Figure 2 presents the proposed rework cycle. It portrays that rework in GSD projects can be detected during work is being done and after work is completed through quality assurance and defect detection activities. Defects can be induced in correct work product (correctly completed work) due to dependencies of correct modules with modules being reworked. This type of rework is coined as *Cascading rework.*

3.1. Workflow of Rework Cycle

In a GSD project, dispersed team members collaborate together to accomplish a *planned work*. While completing the work (indicated in Fig. 2 by *Work in Transit*), quality assurance activities verify whether the work is being performed according to the original specifications and standards. In case of any discrepancy, it has to be reworked and thus, it is transferred to *discovered rework*. Otherwise, after completion, the work would be considered complete.

However, the *completed work* may contain undiscovered errors. Thus, it is composed of *correct work product* and *work product with undiscovered rework*. *Correct work product* is work without errors, whereas *work product with undiscovered rework* is completed work with undetected errors in it. Defect discovery methods discern defects from undiscovered rework. As soon as errors are detected, this *undiscovered rework* will get transformed into *discovered rework* that needs to be again accomplished along with other ongoing works. While performing *rework*, some errors may again be introduced in work being done (*work in transit*) which would either be detected during the work or after the work has been completed.

Furthermore, while correcting the detected errors in discovered rework, changes may be needed in correctly implemented work (indicated by *correct work product*). In this process, some errors may be incorporated in this *correct work product* and some part of it could also turn into *undiscovered rework*. Hence, this transformation of work into rework at three points can occur several times during GSD project. This vicious cycle of work transforming into rework is a fact of life in every software development project, but its recurrence and effects are higher in GSD projects. GSD rework drivers are the factors which influence rework in distributed software projects.

3.2. GSD Rework Drivers

These rework drivers can be categorized into causal factors and process factors which are discussed as follows.

3.2.1. GSD Causal Factors

Geographical, temporal, socio-cultural, linguistic, and

organizational distance act as causal factors for rework in GSD projects.

3.2.1.1. Geographical distance (GD)

When individuals are geographically distant, they seldom have opportunity to communicate face to face and must often depend on synchronous or asynchronous communication channels, such as telephone calls, video conferencing, or emails which is not as effective as face-to-face discussions [17]. It could increase rework as it hampers informal communication, work visibility, coordination, knowledge management, information exchange, and progress monitoring [1, 18, 19].

3.2.1.2. Temporal distance (TD)

It can be imposed by difference in time zones, working hours, holidays, and weekends [2]. Due to temporal difference, individuals can use synchronous communication tools only during temporal overlap and often have to wait for an issue to be resolved, as the relevant person may be unavailable at that time [17]. Thus, it increases feedback time, decreases frequency of synchronous communication, limits coordination and collaboration, and obstructs knowledge sharing [19, 20].

3.2.1.3. Socio-cultural distance (SD)

When individuals from different nations and with diverse backgrounds collaborate, they may get frustrated due to difference in frame of reference, inconsistencies related to usage of terminologies and incongruous work practices [17]. These differences may lead to misunderstandings. Team mates belonging to high-wage economy feel threatened to train and share information with their lower-wage economy counterparts, which in turn leads to "them and us" culture [1].

3.2.1.4. Linguistic distance (LD)

Linguistic difference among team members belonging to different nations hamper formal as well as informal communication and increase the probability of misunderstandings due to differences in accent [1].

3.2.1.5. Organizational distance (OD)

Organizational distance may result into incompatibilities related to processes, standards, tools, and work practices [1]. Diverse process maturity and experience levels can cause misunderstandings and rework [20].

3.2.2. GSD Process Factors

In GSD projects, process factors are facets related to GSD process and project management which, if not contemplated carefully would induce rework. Among these factors, some become active only in GSD projects, whereas others influence rework in every software development project but become more dominant in GSD.

3.2.2.1. Communication, Coordination and Collaboration Management (3CM)

Formal as well as informal communication during software development is required for generating awareness, solving issues, monitoring and coordinating development work. It is also needed for making decisions, creating and maintaining relations [21]. However, inadequate informal discussions, lack of face-to-face meetings, restricted synchronous communication cause difficulties in establishing a unified understanding about remote members as well as project. Communication deficiency also impedes interpersonal relationships, knowledge management, and often leads to coordination breakdown [3, 22].

In GSD projects, coordination issues, such as inadequate shared vision, diverse processes, limited informal interaction, and weak professional as well as social relationships adversely affect task allocation, management of project related knowledge, and process synchronization. Misunderstandings and rework caused due to the aforementioned issues lead to high coordination costs [23]. Varied national as well as corporate cultures, languages, and protocols reduce cohesion, cooperation, and trust. Collaboration within distributed teams requires adequate tool support, trainings, cross-site delegation, and frequent visits of management people to offshore locations [24, 25]. Inadequate communication, coordination, and collaboration mechanism in GSD setting may induce rework. Frequency, timeliness, and tools for communication decide the amount of rework induced in distributed teams [2, 26, 27]. Techniques implemented for coordination and collaboration dictates the cost of delay, clarification, and rework in GSD projects [2].

3.2.2.2. Process Management (PM)

Execution as well as management of GSD projects is more challenging as compared to collocated projects. Moreover, processes and techniques initially designed for collocated projects do not consider the negative impact caused by GSD distances. Ineffective planning, inadequate experience, and negligence of these distances can doom GSD projects to failure. Thus, meticulously planned process and project management strategy is required to avoid rework in these projects [1, 18].

3.2.2.3. Knowledge Management (KM)

Effective management of knowledge is particularly essential in distributed projects. However, geographical and temporal distances restrict knowledge sharing whereas; socio-cultural and organizational distances could induce varied interpretation and obstruct distribution [18]. High attrition rate of offshore members, abrupt shrinking and expansion of teams also hamper knowledge management.

3.2.2.4. Team Management (TM)

Members of GSD project team belong to different nations, may be an employee of different organizations, may have dissimilar native languages, and can be temporally separated. These dissimilarities among the members of same team could crop up misunderstandings, fear, distrust, conflicts, incoherence, and weak personal relations [18].

3.2.2.5. Vendor Capability (VC)

In offshore outsourcing, it is difficult for client to choose and monitor members of vendor team, however, feasibility of vendor can be checked on the basis of CMMI level, prior GSD experience, domain expertise, and professional certifications of team members [1, 18]. In GSD setting, incapable vendor can induce lots of rework.

3.2.2.6. Customer Management (CM)

Customers and developers may have different viewpoints regarding conflicting requirements. Insufficient involvement or unawareness of customer regarding the requirements, and articulation problems may lead to volatile requirements. These issues could generate rework in GSD environment [1, 19].

Classification of GSD Rework Drivers into Causal and Process factors is illustrated in Figure 3. Rework induced due to GSD distances can obstruct successful execution of processes in GSD projects. The following causal factors for rework in GSD cannot be reduced or removed, but their effects can be mitigated by regulating the process factors.

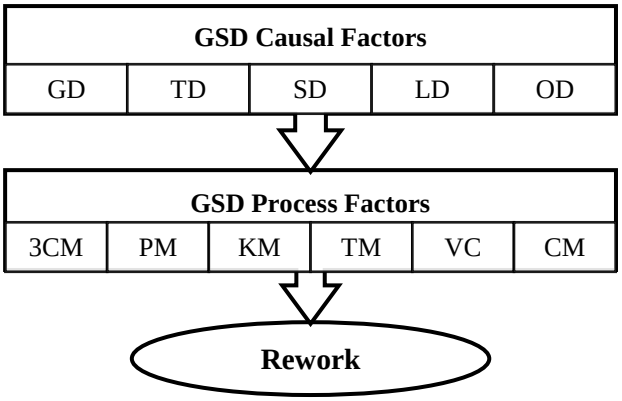

Figure 3 Rework Drivers in GSD

4. COST OF REWORK

Lord Kelvin, a mathematical physicist and engineer said "If you cannot measure, you cannot improve it" is perhaps the best motivation behind using metrics [29]. Rework can substantially increase the effort required to complete a project. We have proposed a metric for calculating additional cost incurred due to rework. The cost of rework can be measured in terms of additional effort expended due to rework.

Rework cost of a module is sum of effort spent for rework in that module and effort needed to modify the dependent modules, which could again provoke rework in their dependent modules (*cascading rework*). Thus, cost of rework in module Mi, Cr (Mi) can be calculated using a recurrence relation as shown below:

$$Cr\ (Mi) = Crm\ (Mi) + \sum_j Cr\ (Mj)$$

Where,

Crm (Mi): Effort spent to perform rework in the module Mi.

Here, *Crm (Mi) = Cri (Mi) + Crw (Mi)*

Cri (Mi): Effort of isolating and analyzing extent of rework for module Mi

Crw (Mi): Effort of implementing rework in module Mi

$\sum_j$ *Cr (Mj)*: Sum of rework cost of all the modules Mj having direct dependencies with module Mi.

The rework cost would depend upon the following rework cost drivers in GSD projects:

- *Dependencies:* Number and strength of dependencies between implemented modules would influence cost of rework. Large number of strongly dependent modules would induce cascading changes, thus increase cascading rework.
- *Recurrent modifications:* Recurrent modifications in a module would rupture the underlying design of module and increase the effort needed for rework.
- *Project configuration:* Size of a user story, complexity of a user story, and underlying technology influence rework cost [29].
- *Quality assurance processes:* Adequate amount of quality assurance processes (such as inspections and reviews) performed during software development would aid in detecting errors early in development cycle, and in turn will reduce the cost of rework.
- *Duration of incremental release:* Very short iterations (for example, 1.5-2.5-week duration) can increase schedule pressure which would pressurize members to complete the tasks without focusing on quality, and consequently induce rework in later iterations. Appropriately sized releases can help in detecting errors early; thus, reduce amount as well as cost of rework [30].
- *Frequency of meetings:* Regular distributed meetings of onshore and offshore members improve project visibility, detect, and avoid misunderstandings timely [18].

5. INDUSTRIAL VALIDATION

The concepts proposed by us are validated with the help of software practitioners. For this, we have performed an industrial survey through questionnaire. In section 5.1, we have discussed design of questionnaire as well as survey and in section 5.2 survey results are discussed.

5.1. Questionnaire Design and Survey

The questionnaire created for industrial survey consists of three sections; namely, introduction, rework, and professional details. The introduction section briefly describes the intention of questionnaire and concisely explains rework. Rework section contains questions which help in the verification of concepts proposed by us. First question is related to verification of factors which influence rework. In question 2, cost drivers for rework in GSD projects are verified. Question 3 is used to verify the concept of rework cycle and cascading rework. Here, respondents are asked to verify whether rework can be cascaded to previously correct and complete dependent modules which could further induce rework in their dependent modules, thus forming a rework cycle. In question 4, their advice was requested about the rework cost metric (summation of two efforts Cri, Crw for calculating cost of rework). At the end of questionnaire, they were asked to fill their personal and professional details such as name, e-mail id, company's name/CMM level, job function, and years of experience in software industry as well as GSD industry.

The questionnaire was created using survey monkey and is verified by two software practitioners and an experienced researcher of GSD domain. The link of questionnaire was sent to software practitioners through email and WhatsApp. In total, 34 responses were received from software industry, which were analyzed to interpret the survey results.

5.2. Result of Industrial Survey

Interesting results have been received from software industry which is analyzed in subsequent subsections. In Section 5.2.1, contextual information of respondents is briefly described. Factors influencing amount of rework in distributed projects are discussed in section 5.2.2. Section 5.2.3 discloses the impact of the cost drivers on cost of rework in GSD projects. Section 5.2.4 presents the views of software professionals about the existence of rework cycle whereas, section 5.2.5 discusses their perception towards the effort components involved in cost of rework.

5.2.1. Contextual Information of Respondents

Contextual information of respondents provides a general picture about the individuals who have filled the questionnaire. Here, a brief summary of contextual information of respondents is presented. 65% of the respondents have software industry experience greater than 9 years, whereas 23% of them have experience of 6-9 years.

Rest 12% respondents have experience between 3.5 to 5 years. Out of these respondents, 44% practitioners are working in GSD projects from last 8 to 14 years, whereas 38% have GSD experience between 3.5 to 7 years and rest of them have at least 2 years of GSD experience. This indicates that respondents are well experienced and capable of providing useful insights for the researched topic. Respondents are working as technical architect, solution architect, project manager, business analyst, team lead, consultant, or test engineer in renowned software companies. Thus, the responses exhibit diverse perspectives for rework in GSD industry.

5.2.2. Impact of Rework Drivers on Amount of Rework

The result of survey reveals that the identified GSD rework drivers increase the amount of rework in GSD as shown in Figure 4. It can be seen that GSD process factors (3CM, process management, knowledge management, team management, vendor capability and customer management) highly influence the rework as compared to causal factors (GSD distances). From the results, it can also be inferred that effect of GSD causal factors can be nullified by improving and establishing effective processes for GSD projects, which can further reduce rework during development activities.

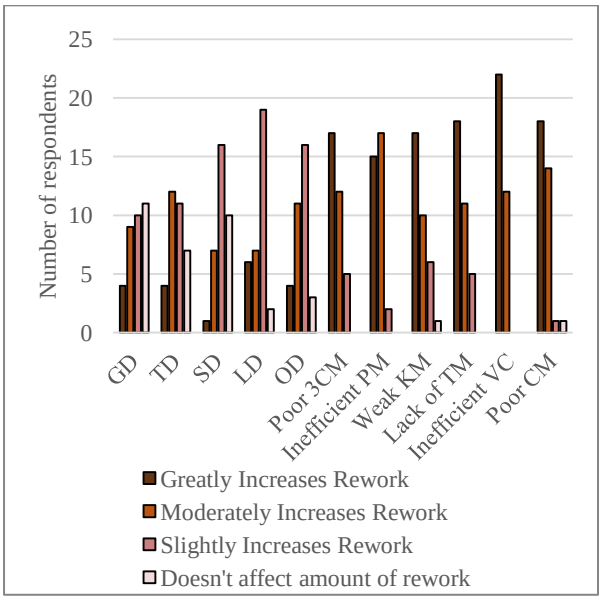

Figure 4 Impact of Rework Drivers on Amount of Rework in GSD

5.2.3. Cost Drivers for Rework in GSD

The result of industrial survey for verification of factors that influence the cost of rework is shown in Figure 5.

Most of the respondents believe that high dependencies, recurrent modifications, complex user stories, as well as complicated underlying technology increase the cost associated with rework. 71% respondents have opinion that large sized story can increase cost of rework in contrast to 27% who believe that size of story doesn't affect cost of rework.

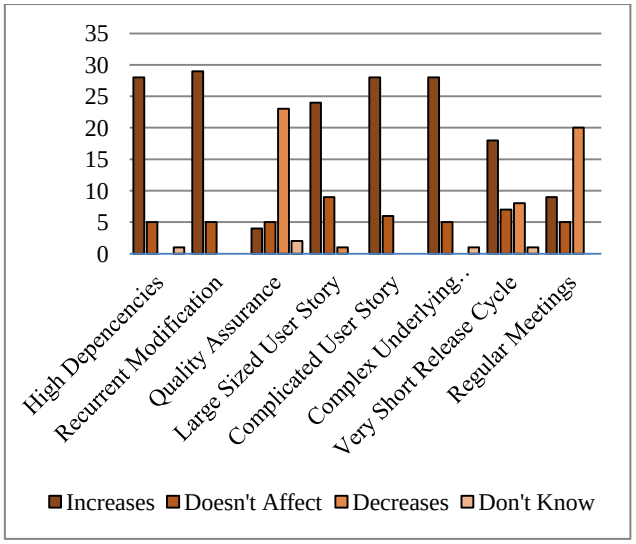

Figure 5 Impact of Factors which Influences Rework Cost in GSD

Respondents of the survey have two contradictory opinions about very short incremental release as 60% respondents believe that it increases rework while 24% respondents believe that it would decrease cost of rework. We have discussed the results of industrial survey with few software practitioners having approximately 13 years of GSD experience. They have disclosed that rework cost would actually depend upon the length of iteration, as iteration of 4 to 6 weeks normally decreases cost of rework as it would promote regular feedbacks from customer and thus, issues would be identified early as compared to traditional waterfall approach. Contradictory to this, if the iteration length squeezes to 3 days to 1.5 week, it could increase rework due to excessive pressure exerted on team members to accomplish the decided work before release.

Similarly, regular meetings can also increase or decrease the rework depending upon their frequency. Daily or twice a day long meeting can increase rework, as developer would get less time for development and a reasonable amount of time would be wasted in the meetings, which would increase work pressure resulting into rework. In contrast, very short daily scrums or alternate day short meetings would maintain their awareness and knowledge about project and aids in timely resolution of issues, which can decrease rework cost. As project awareness and knowledge management are also challenges that are usually faced by GSD practitioners.

5.2.4. Rework Cycle

Rework in some module can be cascaded to previous correctly implemented dependent modules, which could further induce rework in their dependent modules. This can result into vicious rework cycle. Figure 6 reveals that 6% respondents always waste their effort due to rework cycle. It can be interpreted that 53% respondents agree that sometimes it is possible to encounter rework cycle during software development, whereas 32% of respondents advocate that they usually encounter rework cycle whereas 9% respondents have rarely encountered such situation. Thus, it validates the concept of rework cycle and the proposed underlying theory of rework cost metric. Discussion with the software practitioners reveal the fact that the probability as well as extent of cascaded rework can be significantly reduced, if the dependent modules to be affected by rework is identified cautiously; design is modular; dependencies are well

identified; developers are highly competent; and any kind of ambiguities are timely resolved.

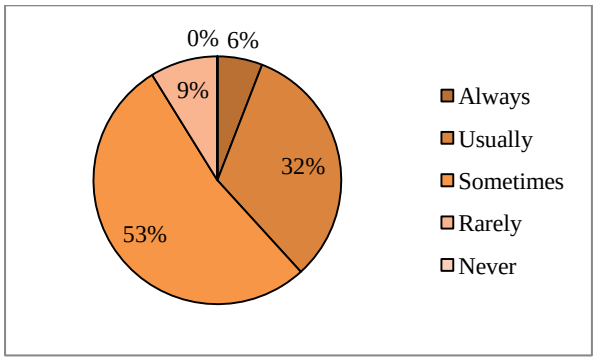

Figure 6 Survey Result for Occurrence of Rework Cycle in GSD

5.2.5. Cost of Rework

Rework cost of a module is sum of effort exerted to isolate and analyze rework and effort for accomplishing rework. Figure 7 illustrates that 88% of respondents either strongly agree or agree with this cost calculation whereas 12% neither agree nor disagree with the concept and none of them disagreed. This result in combination with result discussed in Section 5.2.4 jointly validates the proposed cost metric.

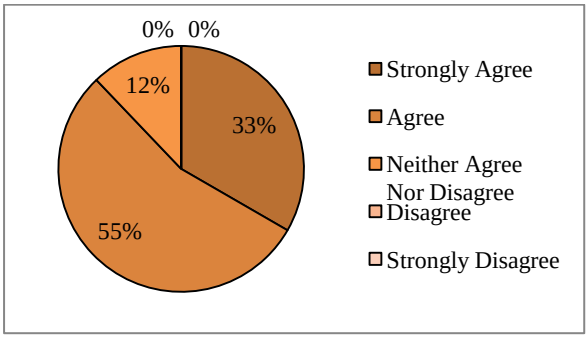

Figure 7 Survey Results for Rework Cost Metric

6. FINAL CONSIDERATIONS

6.1. Discussion

Based on the result of industrial survey, we have derived several inferences.

- GSD rework drivers influence amount of rework in GSD projects. We have categorized these factors into causal factors and process factors. The result of survey indicates that process factors greatly influence rework in distributed projects. However, causal factors interfere with the proper functioning of these process factors in GSD setting. Thus, both these factors directly or indirectly influence rework in distributed projects.
- Cost drivers influence the effort needed to perform rework in a module. High dependencies, recurrent modifications, large and complicated user stories, complex technology, very short release cycle, and very frequent meetings (daily or twice a day) can increase the cost associated with rework, whereas quality assurance activities, optimal sized release cycle, short regular meetings or alternate day meetings can reduce the cost of rework.
- Result of survey verifies that rework in a module can be cascaded to previously implemented modules and can induce rework cycle.
- Survey result verifies that the rework cost of a module is sum of effort spent for rework in that module and effort needed to modify the dependent modules.

6.2. Conclusion

Global software development is a prevailing trend which has fascinated most of the software companies across the world whereas, rework is a dominant software development waste that need to be avoided or reduced to improve the effectiveness of software development. But, in order to avoid and reduce rework, it need to be conceptualized for GSD setting. Measurement of effort associated with rework is also necessary as it also contributes to the overall software development effort. Thus, in this research work, we have investigated the aspects of rework in GSD. A rework cycle is proposed and rework drivers which induce rework in GSD are explored. A metric to calculate cost of rework for GSD projects is also proposed. Cost drivers which can influence rework cost are also identified. The research has been validated through industrial survey. This study could aid in improving the effectiveness of GSD processes by reducing rework. This study would provide a platform for further research in this domain.

7. REFERENCES

[1] R. Jain, U. Suman, A Systematic Literature Review on Global Software Development Life Cycle, ACM SIGSOFT Softw. Eng. Notes. 40 (2) (2015) 1-14.

[2] J.A. Espinosa, E. Carmel, The Impact of Time Separation on Coordination in Global Software Teams: A Conceptual Foundation. Softw. Process Improv. Pract. 8 (4) (2004) 249-266.

[3] M.A. Babar, M. Zahedi, Global Software Development: A Review of the State-Of The-Art, IT University Technical Report Series, IT University of Copenhagen. (2007- 2011).

[4] J.D. Herbsleb, A. Mockus, T.A. Finholt, R.E. Grinter, An Empirical Study of Global Software Development: Distance and Speed, in Proc. IEEE ICSE '01. (2001) 81-90.

[5] B. Boehm, V.R. Basili, Software Defect Reduction Top 10 List, Ieee Software. 34(1)(2001) 135-137.

[6] M.S. Krishnan, The Role of Team Factors in Software Cost and Quality: An Empirical Analysis, Information Technology & People, 11(1) (1998) 20-35.

[7] R.E. Fairley, M.J. Willshire, Iterative Rework: The Good, the Bad, and the Ugly, IEEE Computer. 38(9) (2005) 34-41.

[8] Geneca, Doomed from the Start? Why a Majority of Business and IT Teams Anticipate Their Software Development Projects Will Fail. Winter 2010/2011, Industry Survey (2011).

[9] V. Ramdoo, G. Huzooree, Strategies to Reduce Rework in Software Development on an Organization in Mauritius. International Journal of Software Engineering & Applications. 6(5) (2015) 9-20.

[10] K. Schwalbe, Information Technology Project Management. 8th ed. Cengage Learning, (2015).

[11] C. Deephouse, T. Mukhopadhyay, D.R. Goldenson, M.I. Kellner, Software Processes and Project Performance, J. Manag Inf Syst.12 (3) (1995) 187-205.

[12] T. Sedano, P. Ralph, C. Péraire, Software Development Waste, In Proc. IEEE ICSE '17, (2017) 130-140.

[13] B.W. Boehm, P.N. Papaccio, Understanding and Controlling Software Costs. IEEE Trans. Softw. Eng. 14(10) (1988) 1462-1477.

[14] E. Tonnellier, O. Terrien, Ed., Rework: Models and Metrics An Experience Report at Thales Airborne Systems, ser. Complex Systems Design & Management. Berlin, Heidelberg: Springer. (2012) 119-131.

[15] V.R. Basili, S.E. Condon, K.E.I. Emam, R.B. Hendrick, W. Melo, Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components. In Proc. ICSE' 97, (1997) 282-291.

[16] L. Damm, L. Lundberg, C. Wohlin, A Model for Software Rework Reduction through a Combination of Anomaly Metrics. J. Syst. Softw. 81 (11) (2008), 1968-1982.

[17] D.C. Gumm, Distribution Dimensions in Software Development Projects: A Taxonomy, IEEE Software. 23(5) (2006) 45-51.

[18] R. Jain, U. Suman, A Project Management Framework for Global Software Development, ACM SIGSOFT Softw. Eng. Notes. 43 (1) (2018) 1-10.

[19] R. Jain, U. Suman, An Adaptive Agile Process Model for Global Software Development, IJCSE. 9 (6) (2018) 436-445.

[20] J.D. Herbsleb, Global Software Engineering: The Future of Socio-technical Coordination, In Proc. FOSE'07, (2007) 188-198.

[21] M. Paasivaara, C. Lassenius, J. Pyysiäinen, Communication Patterns and Practices in Software Development Networks. In Proc: 10th International Product Development Management Conference, European Institute for Advanced Studies in Management, Brysseli, (2003) 783-798.

[22] V. Gomes, S. Marczak, Problems? We All Know We Have Them. Do We Have Solutions Too? A Literature Review on Problems and Their Solutions in Global Software Development, in Proc. IEEE ICGSE'12, Porto Alegre, Brazil (2012) 154-158.

[23] P.J. Ågerfalk, B. Fitzgerald, H. Holmström, B. Lings,B. Lundell, EÓ Conchúir, A Framework for Considering Opportunities and Threats in Distributed Software Development. in Proc. International Workshop on Distributed Software Development (2005) 47-61.

[24] M. Bass, J.D. Herbsleb, C. Lescher, Collaboration in Global Software Projects at Siemens: An Experience Report, In Proc. IEEE ICGSE'07 (2007) 33-39.

[25] B. Sengupta, S. Chandra, V. Sinha, A research agenda for distributed software development, In proc. ACM ICSE'06 (2006) 731-740.

[26] H.L. Iacovou, R. Nakatsu, A Risk Profile of Offshore-Outsourced Development Projects, Commun. ACM 51(6) (2008) 89-94.

[27] X. Wang, F. Maurer, R. Morgan, J. Oliveira, Tools for Supporting Distributed Agile Project Planning, Agility Across Time and Space, Springer, Berlin, Heidelberg. (2010) 183-199.

[28] E. Kupiainen, M.V. Mäntylä, J. Itkonen, Using metrics in Agile and Lean Software Development - A Systematic Literature Review of Industrial Studies. Inf. Softw. Technol. 62,C, (2015) 143-163.

[29] R. Baggen, J.P. Correia, K. Schill, J. Visser, Standardized code quality benchmarking for improving software maintainability, Software Qual. J. 20 (2012) 287-307.

[30] A. Gopal, T. Mukhopadhyay, M.S. Krishnan, The role of software processes and communication in offshore software development. Commun. ACM. 45(4) (2002) 193-200.