

Global Software Development Challenges and Techniques for Software Product Quality in Organizations: A Critical Review

Patrick Enabudoso
Walden University

Submitted:

August 27, 2026

Accepted:

September 16, 2026

Revised:

September 30, 2026

Published:

October 16, 2026

Citation:

Enabudoso, P. (2026).
Global Software
Development Challenges
and Techniques for
Software Product Quality
in Organizations: A Critical
Review. *Journal of
Management Policy and
Practice*, 27(2), 103–129.

Leaders in Global Software Development (GSD) organizations have experienced tremendous growth because of increased GSD awareness, greater benefits, and improved process effectiveness in distributed settings. GSD acceptance and adaptation in dynamic, complex environments may be attributed to an added advantage. The primary GSD challenges identified are team coordination, communication, collaboration, trust, culture, people, geographical distance, and time zones. This study reviewed the literature and identified appropriate GSD techniques that enable leaders to implement best management practices to improve software product quality. This paper provides a detailed critical literature review, summarized results, and a comparative study. This study identified emerging GSD challenges in the literature as potential success factors when properly addressed. In addition, the research confirmed GSD techniques that may help leaders manage software product quality effectively, avoid project failures, and meet customer expectations.

Keywords: global software development, leaders, organization, software product quality, challenges, techniques, success factors

INTRODUCTION

Leaders in organizations have begun exploring standardized strategies to produce reliable, economical, and safe software for human benefit. Global software development easily creates quality problems that require alignment among all stakeholders (Barney et al., 2013). Software requirements are highly complex, especially as markets grow and more aspects of life require enhanced software products. Some companies adopted cheap software design and programs to meet market demands and supply. The result is poor-quality software that can harm the public and organizations. GSD leaders need to understand the benefits of building quality into software production at every stage of the software development life cycle (Enabudoso & Schuttler, 2026). The concept of organization management continues to evolve with new ideas and emerging techniques to solve software

development problems, improve the standard of living, and ensure public safety. Software product quality is mainly a managerial problem. Leaders can manage a GSD team with skills in collaboration, communication, and cultural understanding (Wickramarachchi & Lai, 2016). The most persistent challenges of software failure in GSD are attributed to a poor-quality culture in software engineering (Usman & Khan, 2018). Quality is defined as the degree to which an in-built set of characteristics meets a product's standard requirements (PMI, 2018). These mandatory quality attributes affect the processes and services an organization provides. GSD is offshore software development with peculiar characteristics in a distributed environment different from a collocated or in-house software development setting (Chadli et al., 2016; Rehman & Khan, 2014). The dynamic and growing GSD trend offers benefits, time savings, cost-effectiveness, and competitive advantages over traditional software development methods (Enabudoso, 2020). Ryan (2016) argues that the most effective software development approaches face challenges from internal and external factors. The outsourced method is described as breaking cultural and geographical barriers, but it also presents numerous challenges (Usman & Khan, 2018). The ultimate challenge of GSD is identified as the quality of software products for customer acceptance (Rehman & Khan, 2014; Usman & Khan, 2018), as cited by Enabudoso (2020). This research aims to explore GSD challenges and identify techniques in the literature that could improve the software development process in geographically distributed settings. This paper is organized into five sections. The background of GSD challenges, a summarized literature review with the GSD Challenges and identified Techniques. The prospective future work and conclusion. Literature Search Strategy: The literature review strategy involved a focused review of studies published in peer-reviewed articles and scholarly journals. It also included materials from institutions and organizations, including industry and government reports, conference proceedings, and books. The information gathered led to a comprehensive evaluation of GSD challenges in organizations and their effects on software product quality. Insights came from studying GSD methods, current practices, and suggested future studies. To find relevant information, the researcher used Internet search engines and relevant research data sources, including Walden University Library, Google Scholar, IEEE Xplore, Elsevier ScienceDirect, EBSCOhost, ScienceDirect, SpringerLink and Wiley InterScience. These sources enabled a comprehensive evaluation of GSD challenges and techniques in organizations and their impact on software product quality.

BACKGROUND

Leaders in organizations face challenges in global software development in a distributed environment. GSD teams often identify problems with communication, collaboration, cultural differences, time zones, requirements and regulations, and technical ability that affect software product quality (Ali & Lai, 2017). Beyond traditional software development methods, companies are adopting techniques such as project management principles, Lean, Six Sigma, Kanban, and Agile methodologies to overcome challenges and remain competitive. Software development in other multi-location environments is growing rapidly because of its benefits and fast-emerging large markets (Enabudoso & Schuttler, 2026).

Every facet of human life is going digital: e-commerce, internet networking, and self-applications require reliable software to meet users' needs. The rapid growth of software development is complex and creates problems in identifying, evaluating, and measuring software quality (Miguel et al., 2014). Software flaws and failures have extreme negative impacts on the cost of achieving business objectives in reputable organizations and major world economies. According to Hossain (2018), the National Institute of Standards and Technology estimates that software failure costs in the U.S. economy are about \$60 billion per annum, or 0.6% of gross domestic product. A lack of customer satisfaction, reliability, and maintainability affects both software development organizations and end users (Hussain, Farid, & Mumtaz, 2019). Globalization is surging and has major impacts on the economy and software development companies. Innovation, Science and Economic Development (ISED) Canada reported that industry structure evaluation of ICT companies by sub-sector showed that software and computer services accounted for 90% of the total (ISED, 2018). GDP by ICT sub-sector totalled \$87 billion. Software and computer services accounted for 48%. Software and computer services achieved a yearly growth rate of 6%, compared to other sectors.

Yaseen, Ali, and Ullah (2016) found that a lack of common understanding of software development problems has led to low software product quality and failure to achieve customer satisfaction. Software developers spend about 80% of developing costs on identifying defects and performing isolations (Rustambekovicha, Gulyamovb, Usmanovac, & Mirzaevd, 2017). This gap in the literature poses a major challenge for GSD. Researchers have identified critical shortcomings. Research on the impacts of GSD is scarce and has limited empirical evidence (Jabangwe et al., 2014; Wickramaarachchi & Lai, Niazi, Mahmood, Alshayeb, & Hroub, 2016). Chadli et al. (2016) reported a lack of advanced, reliable quality models to evaluate and measure the impact of defects on software product quality. Evaluating and predicting quality in a global environment is still at an elementary stage (K. Khan, Khan, Aamir, Zulfikar, & Khan, 2013). Software engineering is considered a new concept, with design, development, and maintenance stages (Gheorghe, 2015). Contemporary Leaders in organizations are working to provide solutions to problems in GSD distributed teams. Other identified GSD challenges prevalent in organizations include: human resources allocation (Bhatti & Ahsan, 2016), customer requirements (Usman & Khan, 2018), and trust and coordination (Nagappan, Murphy, and Basili, 2008). New trends in GSD challenges include global security, governance, government policies, politics, people, culture, and languages.

CRITICAL LITERATURE REVIEW

Origin and Evolution of Global Software Development

Global Software development, software engineering and computer programming can be traced back to the early 1950s, as concepts evolved through industries and institutions. Cusick, Prasad, and Tepfenlart (2008) noted that the science of computing started in the United States and the United Kingdom. Geographically distributed multi-site software development emerged in the 1960s (Carmel, 1999). Organization leaders and researchers faced developmental challenges and shortcomings. The NATO-sponsored conference of 1968 in Garmisch-Partenkirchen raised awareness through research presentations on common software development problems and fostered solutions and recommendations (Ebert, 2018). Software development became a global affair because of software market demand and after the ICT revolution of the 1990s. Khan, Keung, and Al-Wadud (2017) confirmed in their study that modern GSD practices became popular and were adopted because of their benefits and advantages over the traditional collocated approach to software development in industries, beginning in the 1990s. About 200 Fortune 500 companies benefited from GSD firms in 2000, as reported by the Association of Software Service Companies (NASSCOM) in India (Denny, Mani, Nadella, Swaminathan, & Samadal, 2008), as cited by Enabudoso (2020). Software development across multiple sites began in the late 1990s and continued through the mid-2000s, with growing acceptance among software companies (Akbar, Khan, & Adnan, 2020).

Growth, Acceptance and Benefits of Global Software Development

Companies are adopting software development in other distributed geographical environments, without regard for barriers to software quality. Evidence shows a steady transformation of GSD driven by business needs, organizational demand, and community acceptance (Babar & Lescher, 2014). Denhere, Horne, and Van der Poll (2015) noted that software companies set up subsidiaries overseas in developing economies, such as the Middle East, as destinations for software development projects and maintenance. GSD is associated with different names or terminologies. Denhere et al (2015) called GSD *distributed multi-site software development*. Babar and Lescher (2014) derived and used the term *Global Software Engineering*. S.U. Khan and Azeem (2014) coined the term GSD, using the phrase "*Offshore Software Development Outsourcing*," to represent GSD in a study of intercultural challenges in geographically distant outsourcing locations in India, China, Ireland, and Russia. The terms *client and vendor organization* are commonly used in countries like the United States, the United Kingdom, and Japan.

The era of high-speed, reliable internet, ICT, and better telecommunications has enhanced the management of a global virtual team. The collocated team is considered easier to set up and manage than an outsourced GSD team. Researchers and Practitioners identified GSD challenges from the outset, such as

risk, culture, language, religion, quality issues, work ethics, tools, compatibility, time zones, temporary distance, and coordination and communication (Denhere et al., 2015). GSD problems continue to unfold because of the dynamics and complexities of global trade conventions. New barriers could include new technologies, modern business implementation techniques, government policies, politics, cyber-security, regional trade interests, regulations, and standardization. Empirical research results could help leaders improve performance in a virtual team (Iftikhar, Alam, Musa, & Suud, 2018). Combined efforts, such as team integration, may help meet changing demand, functions, and quality requirements (Goltz et al., 2015).

Globalization has compelled organizations to adopt GSD software development strategies that are cheaper, faster, and meet customers' quality requirements (Marinho, Luna, & Beecham, 2018). The GSD method may satisfy investors' needs but requires teams from different cultures to work together, creating potential barriers to communication and coordination (Babar & Lescher, 2014; Yue, Ahmed, Wang, & Redmiles, 2019). GSD growth, benefits, and adoption increased because of available skilled resources, emerging markets, round-the-clock development, integrated capitalization, proximity to markets, and global acquisition opportunities (Yassen & Ali, 2019). One advantage of a multi-site software development team is knowledge transfer and team agility. Huisman and Rubin (2018) agreed that software development provides innovation, learning and knowledge transfer. GSD teams use managerial strategies, advanced technologies, and globally distributed settings to deliver high-quality software while saving time and reducing costs (C. Kahlil & Kahlil, 2019). GSD offers numerous benefits, despite challenges of software product quality (Hidayati, Purwandari, Budiardio, & Solichah, 2018). The continuous evolution of the GSD process benefits organizations and business communities (Babar & Lescher, 2014).

IDENTIFIED GLOBAL SOFTWARE DEVELOPMENT CHALLENGES

GSD challenges affect software quality (Shanyour & Qusef, 2018). Despite the benefits of GSD, challenges persist as technologies evolve and trends become increasingly heterogeneous (Shanyour & Qusef, 2018). The GSD challenges documented in the existing literature continue to increase, and as decentralized software development evolves, new challenges are emerging (Qureshi, 2014; S.U. Khan & Niazi, 2012). These GSD challenges range across several success factors, including human factors (Shameem, Chandra, Kumar, & Kumar, 2018). The people aspect of this challenge refers to the ability to manage people in the workplace to deliver organizational objectives. Managers and leaders have to be people-oriented, growth-driven, motivating, inspiring, and team-building. Poor quality (Jaffar, Hussain, & Chiad, 2019): poor quality culture leads to defects and rework. Technical, political, and social (Suman & Jain, 2015). The GSD team's competency is a critical concern. Political and social perspectives are now a concern because globalization and world leaders influence major economic decisions and global trade. Temporary distance and speed (Carrillo de Gea, Nicolas, Fernandez-Aleman, Toval, & Idris, 2016): Distance and speed in software development delivery are vital issues that multi-site teams must address. Standardization (Jain & Suman, 2015): GSD functional guidelines, procedures, and approaches require common regulation and standardization to maintain ethical values and ensure consistency in products, services, processes, and software quality. Stakeholder (Colomo-Palacios, Soto-Acosta, García-Peñalvo & García-Crespo, 2012). Stakeholder management is imperative to GSD success because all parties must work with the same awareness and agreement. Trust (Trainer & Remiles, 2018): Trust is key to collaboration and effective communication within the team. Risk (Aslam, Ijaz, Lali, & Mehmood, 2017): Risk management is important to GSD because, when not addressed properly, it could lead to project failures. Substituted legacy systems (Matthiesen & Bjorn, 2015); the aspect of replacing legacy systems amounts to effective change management at every stage of the GSD project life cycle. These GSD challenges arise from the dynamic complexity of the modern business market environment.

GLOBAL SOFTWARE DEVELOPMENT TECHNIQUES

Models and Frameworks in GSD

Global software development leaders and end users are concerned with software quality to meet people's needs. The quest for software product quality gave rise to different software development techniques. GSD organizations face the challenge of measuring and evaluating quality. After 2000, using models became imperative for predicting quality. Using models is an acceptable technique to enhance quality management in software products (Miguel et al., 2014). The software quality model has characteristics and relationships that help define quality requirements and evaluation (ISO/IEC 9126-1, 2001). Each quality characteristic includes measurable attributes that enable evaluation of software quality (Tiwari & Chakraborty, 2015). The quality model allowed quality to be built into software by defining standard coding and defect detection in the software development process (Kumar, 2017). Djouab and Bari (2016) adopted a framework dependent on e-learning patterns in relation to the ISO 9126 procedure. Their research findings enabled the evaluation of existing systems. As quality models have evolved, they are classified as basic or tailored. Basic models were derived before 2000, whereas tailored component quality models were developed after 2000 (Miguel et al., 2014).

Adopting ISO/IEC 25010 for software product quality evaluation adds more quality characteristics. Calabrese et al. (2018) identified software security characteristics as a case study, using a defined matrix and results from three studies in which software security was treated as an effect of software product quality. They evaluated software quality alongside GQM (Goal, Question, and Metric), with ISO/IEC 25010 standards. Research findings showed that poor software quality can lead to dire consequences, including loss of competitive advantage, damage to corporate image, monetary losses, and high failure rates (Calabrese et al., 2018). Software development leaders have strived to combine several models to ensure software quality. Alrawashdeh, Muhairat, and Althunibat (2013) derived a quality model for Enterprise Resource Planning (ERP) that adopted the ISO 9126 approach. Polillo (2011) investigated QM methodology in a Web 2.0 site environment, comparing ISO/IEC 9126 and ISO 25010 for software products. QM is considered useful for requirements gathering, evaluation, quality definition, design, process development, operation, and maintenance. Bourgroun, Sabre, and Bouchentouf (2017) conducted an IT assessment to improve software product quality and maintenance costs using Object-Oriented (OO) program model properties. According to Baquero et al. (2018), quality conception or ideas refers to a set of characteristics, artifacts, processes, resources and built to meet requirements or needs. A software quality model could be improved through literature reviews, interviews, questionnaire data analysis, observations, and weighted factors (Sugiyanto, Rochimah, & Sarwosri, 2016). Studies have shown that improved models and frameworks can help GSD leaders alleviate the challenges of software development in a distributed environment.

Process Improvement Management

Organizations adopting GSD face globalization-related cultural limitations and geographical distance as imminent challenges (A. Khan et al., 2017). Virtual software development offers cost-effectiveness, improved software product quality, and return on investment. Software development practitioners face Software Process Improvement (SPI) challenges. Researchers are working to improve GSD processes because distributed software development leads to poor software quality. Investigating the effectiveness of the Software Process Improvement Implementation and Management Model (SPIIMM) is crucial because software process improvement in software development presents significant challenges (A.A. Khan et al., 2017). A systematic literature review was implemented to identify critical success factors, critical barriers, and SPI. The method used survey questionnaires from 111 participants who were SPI experts. The study included input on essential success factors and crucial barriers in the capacity maturity integrated model to assess companies' SPI readiness level and the potency of the SPIIMM framework. A.A. Khan, Keung, Niazi, Hussain, and Ahmad (2017) confirmed the problem that SPI initiatives in GSD companies face. They checked barriers that can negatively affect SPI in GSD organizations from both vendor and client

perspectives, using a literature review and survey questionnaires to identify and validate them. They concluded that identifying SPI barriers can help GSD organizations succeed in projects.

The effectiveness of existing process improvement techniques in a distributed software development setting is critical. Bhatti and Ahsan (2017) agreed that one of the major challenges of GSD is monitoring and controlling. The researchers developed a process improvement framework using grounded theory methodology and a reliable validation technique. The findings indicated that the framework was valid for the GSD geographical locations (Bhatti & Ahsan, 2017). According to Bhatti and Ahsan, GSD leaders who implement process improvement are likely to achieve software quality. They recommended implementing the framework in a distributed environment using both failure and success factors (Bhatti & Ahsan, 2017). Software quality is a main determinant of customer satisfaction (Enabudoso & Schuttler, 2026). Modern GSD trends include implementing a software process model to improve software quality, setting tasks, forecasting timelines, encouraging leadership participation, and supporting a culture of continuous process improvement (Akbar et al., 2018). Further study on improving SPI identified similar success factors using the Fuzzy AHP method to collect responses and compare factors (A.A. Khan, Shameem, Kumar, Hussain, & Yan, 2019). The approach depends on effectively prioritizing success factors to reduce uncertainty during decision-making. The study also identified process improvement techniques, including process and systems, QMS, TQM, CMMI, ISO, SCRUM, Six Sigma, Lean, and Agile methodology. Other high-level models are key to GSD success if implemented properly. These include the Control Objective for Information Technologies and the Portfolio Programmed Project Management Maturity Model (Bhatti & Ahsan, 2017). These models may have been proven effective and efficient.

Requirement Management

Requirement prioritization in GSD organizations is a serious consideration in the SDLC for achieving software product quality (Jayatilleke, Lai, & Reed, 2018). Engineering requirement prioritization involves ranking requirements by value criteria for the release stage (Chimagus, Selamat, Ibrahim, & Mahin, 2014). Software requirements change often, and the wrong implementation can lead to rework, rejection, cost, and wasted time. Software errors can result from improper changes, incorrect fixes, and inaccurate updates (Yi, Qi, Tan, & Roychoudhury, 2015). Requirements change at every stage of the software development life cycle and can stem from customer requests, business changes, or changes in operating environments (Jayatilleke, Lai, & Reed, 2018). Most organizations use change identification and evaluation tools to prioritize requirements. Some organizations develop a framework to allocate requirements according to the project schedule. An incremental approach is sometimes preferable to avoid pitfalls by team members. Awais (2016) used the ETVZ prioritization model, applicable to both in-house and GSD requirement selection. Awais identified the model's challenges and areas for improvement to make it more effective for GSD requirement prioritization. The study's methodology was a literature review of various requirement prioritization techniques and their challenges; requirement types were modularized into subsections to support decision-making, with product values and preferences assigned to the main requirement hierarchy according to customers' priority needs (Awais, 2016). The complexity of GSD stems from change and the need for requirement prioritization analysis (Jayatilleke et al., 2018). A study on a requirements analysis approach analyzed changes by function, identified change barriers, and mapped dependencies using a matrix. Future work may extend this approach to large databases, where the outcome will help decision-makers with requirements and prevent wasted time (Jayatilleke et al., 2018). Most businesses face risk and evaluation; GSD is not risk-free, despite modern techniques and models.

The success of GSD projects depends on effective requirement engineering prioritization and managing challenges, including socio-cultural differences, language barriers, communication, change management, knowledge sharing, and lack of trust. Advanced technologies that aid virtual communication, computer systems, and skilled software engineers offer advantages in GSD (Usmani, Hassan, & Mahmood, 2017). Removing impediments during ongoing requirement updates across all phases of the software development lifecycle is key to delivering a quality product. Usmani et al. (2017) used the requirement improvement model, along with SLR and questionnaires, to gather stakeholder data on the challenges they faced during requirements gathering in a GSD situation. The focus was on how the GSD organizational leaders manage

the challenges of requirement gathering for software development. The research found that organizational leaders are reluctant to bring in requirements engineering for fear of project failure. Requirements engineering is necessary early on, using available tools to make the best of the GSD setting (Usmani et al., 2017). Requirement Change Management (RCM) is an essential part of the SDLC in the GSD methodology because change is inevitable but can be managed (Anwer, Wen, Wang, & Mahmood, 2019). Studies consider the following requirement stages: elicitation, specification, change management, validation, and documentation (Ali & Lai, 2017; Anwer et al., 2019; Awais, 2016). Anwer et al. (2019) adopted a requirement change management approach similar to Usmani et al. (2017). They indicated that traditional methods of validating and documenting requirements for in-house software development may not be effective in the GSD environment.

Integration Management (Communication and Collaboration)

Software systems have become essential to business transactions worldwide. Communication is vital during requirements gathering, as well as during coordination and control of virtual teams in different locations. Communication risk is more glaring during requirements change management because changes occur frequently during the SDLC. The root causes of software failure in GSD are referred to as communication, collaboration, and control (A.A. Khan, Basri, Dominic, & Amin, 2013); this requires managerial attention to guide the team. Software product quality issues are most common when managing outsourcing relationships, which can lead to low-quality products (Guru Anz & Iyortsuun, 2017). Communication risks associated with GSD can arise from internal or external threats due to interactions beyond socio-cultural, geographical, and temporal boundaries (Rehman & Khan, 2014). Building trust among GSD stakeholders across different environments is important for team integration from the outset of a project (Arumugam & Kaliamourthy, 2016). Integration management is key to team performance and success. Sharma, Kaur, and Kaur (2015) noted that agile software development may benefit from GSD because agile enables collaboration, interaction, continuous requirement gathering, cost savings, and prompt software project delivery. Sharma et al. (2015) investigated challenges in the literature and industry and developed a framework to help alleviate GSD communication problems. After a systematic literature review and industry study, the authors concluded that implementing the agile method in a distributed environment can mitigate challenges with appropriate tool support (Sharma et al., 2015). Previous research and current work highlight both benefits and challenges. A language processing model was used to address the language-translation problem in GSD settings (Sharma et al., 2015). Horta et al. (2019) argued that most customers demand high-quality software and requested experienced software managers who can interface with team members so they can collaborate and solve complex problems. Stack Overflow was termed a useful model network, and NetSCAN algorithms were used to identify integrated teams (Horta et al., 2019). Findings showed that high-performing teams were motivated.

A good understanding of integration failure may help develop an appreciation of most integration challenges in the GSD environment (Zafar et al., 2018). Leaders reported that controlled integration, often overlooked during development, can suddenly become a problem during the system integration phase because of incompatibilities. Zafar et al. (2018) examined integration success factors through an industrial survey, literature review, concept mapping, and the development of a taxonomy of 40 integration failure factors. They found that evaluating the taxonomy of failure factors enabled identification of the integration problem domain at the early stage of the project (Zafar et al., 2018). Proper integration in a distributed environment would enhance communication

Knowledge Transfer Management

GSD organizational leaders strive to comply with knowledge management standards and practices that help virtual team members understand and share knowledge. Software production in an outsourced location is knowledge-intensive (Razzak, Bhuiyan, & Ahmed, 2014). Effective team knowledge management ensures product quality, social acceptance, and usefulness (Ouriques, Wnuk, Gorschek, & Svensson, 2019). Knowledge consists of the data and information teams need to perform their tasks. Prevalent collaboration and communication problems in distributed software development hindered knowledge sharing and

documentation, leading to knowledge vaporization (Waheed et al., 2019). Additional knowledge management efforts, tools, and techniques are needed to share information and knowledge among team members (Huzita et al., 2012). Effective knowledge management is the backbone of software development (Ahmed, Mansor, & Ahmad, 2017). A lack of team knowledge can lead to inconsistency, low productivity, poor documentation, and low-quality software development (Niazi et al., 2016). GSD organizational leaders focus on effectively managing and sharing knowledge among team members because of the perceived benefits. Sometimes, knowledge resides within team members, which in turn positively impacts processes, methodologies, the environment, and company assets (Waheed et al., 2019). Knowledge management makes it easier to share, distribute, create, and capture the knowledge needed to build a software product that meets product and client expectations (Ahmed et al., 2017). Team effectiveness is improved through knowledge sharing and mutual understanding. Knowledge sharing includes all stages of the software development life cycle. Researchers are investigating elements of knowledge management: purposes, knowledge types, and technologies to achieve high software quality for end users. Knowledge reuse among team members is now gaining more attention in distributed software development environments (Indumini & Vasanthapriyan, 2018).

Researchers are studying knowledge management effectiveness in the software development environment using different methods. Ahmed et al. (2017) examined agile problems in software development, informal tacit knowledge and GSD explicit knowledge, and formal communication modes in a dispersed environment across time and space; challenges in the literature, based on frequency of occurrence, pertained to agile development with GSD. Ahmed et al. (2017) identified the most critical challenges as managing knowledge storage, knowledge sharing, knowledge transfer, and tacit knowledge. Additional findings included knowledge creation, a lack of a common universal language, and minimal impact from documentation (Ahmed et al., 2017). Future knowledge processes require verification, validation, specialized methodologies, and matching newly discovered challenges against identified existing challenges (Ahmed et al., 2017). Collaboration among team members is a major issue for knowledge sharing in a distributed space. Waheed et al. (2019) developed a scheme with documented records and knowledge vaporization issues as inputs, then evaluated it using an industrial case study. Waheed et al. (2019) examined the framework using real-life scenarios; data came from interviews, analyzed using thematic analysis, and a System Usability Scale (SUS) questionnaire. The research included team members confirming that the proposed solution improved the knowledge-sharing process (Waheed et al., 2019). Contextual information management required more attention because GSD virtual teams need extra effort in coordination and communications (Huzita et al., 2012). Areas of concern included challenges and opportunities for conceptual information management. The research considered how to manage knowledge, performance, socio-cultural factors, information storage, and data sharing in GSD (Huzita et al., 2012). The research examined how GSD organizational leaders would understand and use a mechanism to capture contextual ideas using the cloud computing paradigm. Physical distance in a distributed environment challenges effective collaboration among team members, primarily affecting cost and quality (Huzita et al., 2012). Contextual information refers to team members knowing how information came about and how to use it. GSD organizational leaders expect to collect, store, process, and distribute information to the team. A distributed software environment needs the right infrastructure and mechanisms to capture contextual ideas for knowledge sharing (Huzita et al., 2012).

Tasks Allocation Management

GSD leaders are trying new strategies to overcome problems and risks. Planning and managing task allocation in a development structure is considered critical and challenging (Mahmood et al., 2017). Improper criteria for task allocation decisions at the planning stage are reported to complicate the process (Filho, Pinheiro, & Albuquerque, 2015). Organizational leaders use prioritization and task classification during planning, and task assignment is a critical management activity throughout the software development life cycle and requires informed decisions (Filho et al., 2015). Improper task allocation can significantly affect the project schedule, cost, and quality. Task allocation and prioritization are important aspects of software development. Filho, Pinheiro, and Albuquerque (2018) examined a survey to evaluate

task allocation in GSD projects using an agile approach. The methodology was a systematic literature review of task allocation in distributed environments and task allocation using agile methods. The research found that GSD in agile methods is limited but included prioritization and classification of task allocation. The authors recommended using a multi-criteria model to support the prioritization and classification of task allocation to improve team performance. GSD researchers proposed several models and frameworks to address the complexities of task allocation. Few studies have examined the criteria influencing task allocation in a socio-culturally and geographically impacted dispersed workforce. Filho et al. (2018) progressed the research on task allocation by using multicriteria models to allocate tasks in a GSD project. Findings showed that the approach would guide researchers pursuing future studies on task allocation in distributed software development settings (Filho et al., 2018). Recommended that future studies may identify and address the impact of factors influencing task allocation.

GSD leaders are expected to be familiar with task-allocation criteria at every stage of the SDLC. Mahmood et al. (2017) used a similar approach of an SLR and a questionnaire survey as did Filho et al. (2018). Feedback came from 62 industry practitioners; results showed no significant difference between the SLR outcome and the questionnaire results (Mahmood et al., 2017). Key factors included were task size and cost in a decentralized GSD structure. Leaders should understand the factors influencing task-allocation criteria at the planning stage to make appropriate task decisions (Mahmood et al., 2017). Mahmood et al. (2017) identify practitioner-proposed task allocation techniques, including the sun-development model, cost model, and risk-based multi-criteria optimization model.

Project Management

GSD includes client and vendor organizations, virtual teams, enabling technologies, and other resources. Software quality can be the strongest or weakest link in the software development process (Birgun & Altan, 2019). Hence, effective project management is vital for meaningful transformation in GSD. Software development and use are now part of the digital business environment in every community, whether through mobile devices, the web, or computer systems (Rajagopalan, 2014). Current digital evolution is evident in Industry 4.0 structure: robots, smart factories, cybersecurity, cloud computing, and integrated systems that rely on software (Birgun & Altan, 2019). Contemporary software managers in organizations are increasingly shifting to GSD because of its benefits of high quality and low development costs (Niazi et al., 2016). Managing software development processes, organizational interfaces, and interactions among geographically distributed team members requires leadership. Software development leaders must ensure excellent communication, coordination, and control strategy of the virtual team to achieve success. The main stages of the SDLC include planning, creating, testing, and deployment (Magableh & AlSobe, 2018). A good, realistic management plan helps manage distributed software development effectively because of perceived hidden costs, coordination and communication challenges, and a lack of GSD project management readiness (Niazi et al., 2016). Khraiweh (2013) argued that software development planning entails resource analysis, work breakdown, timing, clear direction, derived deliverables, defined responsibilities, and identified project constraints. Proper planning and documentation are key success factors in software development quality (Denhere et al., 2015; Rajagopalan, 2014). The project life cycle is termed a series of phases from project start to completion (PMI, 2018). Project life cycles could be either predictive or adaptive. Project management is a vital key to software development (Rajagopalan, 2014). Beyond project planning, other essential project management processes include initiating, planning, executing, monitoring, and closing (PMI, 2018). In the PMI guide on project management, these processes are organized into knowledge areas: integration, resource, scope, risk, schedule, flexibility, cost, quality, communication, procurement, and stakeholder (PMI, 2018). GSD organizational leaders can use these management group processes and categorized knowledge areas, depending on project needs, to initiate and complete a software project on time.

GSD organizational leaders continually explore alternative strategies for remote software development because of many advantages, including proximity to market, skilled workers, all-round development across different time zones, and the availability of advanced technologies that aid good communication and coordination (Saleem et al., 2019). Adopting project management processes is necessary for the success of

software development in a distributed GSD setting. Some benefits of remote software development include favorable government tax policies, lower development costs, and improved quality and regulation. Destination concerns include infrastructure, communication connectivity issues, and the coordination of virtual team members (M. Khan & Khan, 2018). GSD organizational leaders need to understand how to manage a virtual team better despite the popularity of available communication technology tools. Changes in technology versions and test labs could pose a problem for team members (M. Khan & Khan, 2018). M. Khan and Khan (2018) developed a team management challenge mitigation model to assist with virtual team management. The authors used an SLR method to identify challenges encountered by outsourced vendors and used industry questionnaire surveys to validate the SLR findings (M. Khan & Khan, 2018). They examined project management success factors in GSD through an SLR, then compared them with a set of collected questions from software development participants in the industry. M. Khan and Khan (2018) derived project management success factors as project managers, organizational structure, culture, and trust. The research found that the model could help predict and measure company readiness for project management in GSD settings (M. Khan & Khan, 2018).

Effort and Cost Estimations

GSD organizational leaders build trust in effort estimates for a software project, which is an integral part of project management. Cost estimation predicts the effort required by software managers from the start to the end of a project (Keshta, 2017). According to Ramacharan and Rao (2016), software managers prefer GSD because it offers a shorter software development cycle. Effort estimation is the preliminary phase of the software development process, where project definitions are established with a certain level of estimation accuracy. The project charter authorizes the project before the planning, executing, monitoring, and closing phases begin (PMI, 2018). Effort cost estimation is useful in software development project processes, serving as input for iterative project planning, contract bidding analysis, resource allocation, procurement, and financial budget evaluation (Keshta, 2017; Ramacharan & Rao, 2016). The study used two empirical search-based models (COCOMO II and SLIM Models) (Ramacharan & Rao, 2016).

The schedule-based estimation model with calibration parameters provided evidence of accurate effort estimation (Ramacharan & Rao, 2016). Researchers used COSEKMO in the methodology, a software tool for making selections. Both models provided similar accuracy for in-house software development and GSD projects. The schedule-based model was more accurate for GSD projects. The recommendation was that future researchers should compare model-based and GSD expert-based approaches (Ramacharan & Rao, 2016). Keshta (2017) used the same class of software development estimation models (COCOMO II and SLIM Models) in an investigation and compared them with four other estimation models: COCOMO, Hybrid, Putnam, Function Point, and Wavelet Neural Network. The findings indicated that project managers should use more than one estimation model so they can properly evaluate results before making final decisions. Boehm (2017) argued that one-size-fits-all software cost-schedule estimation models are no longer ideal due to the GSD evolutionary pattern of wide variations and features. Software estimation is a continuous process from project initiation through the development life cycle and a key success factor in project management (Keshta, 2017).

Organizational Structure

GSD management requires organizational leadership to ensure effective coordination, communication, and control among team members. Managerial difficulties and concerns accounted for 70% of GSD-reported communication problems (Ammad, Janjua, Madni, Cheema, & Shahid, 2017). Despite the emergence of advanced new telecommunication technologies, collaborative environments, and group support systems, managing a virtual team remains a major challenge (Shameem, Kumar, & Chandra, 2017). Enforcing applicable GSD countermeasures would be ineffective without addressing the managerial aspects of GSD in distributed locations (Ammad et al., 2017; Birgun & Altan, 2019). Project performance in in-house software development is higher than in GSD because GSD lacks communication and coordination. Galin (2018) identified procedures that ensure timely performance in software projects, including implementing project control components such as monitoring schedules, budgets, and quality. Globalization

affects the economy and, in turn, the software industry. Project managers can analyze all aspects of project performance through comprehensive feedback to evaluate effectiveness.

Top management sets quality in the boardroom and has the power to make decisions and changes to design or systems (Deming, 2013). GSD management challenges affect product quality and the stakeholder engagement process (Ammad et al., 2017). GSD methods have inherent issues: management, technical, organizational, team, and customer concerns that affect communication among multi-site team members (Alzoubi, Gill, & Al-Ani, 2016; A. Khan, Keung, Hussain, & Bennin, 2015). The increasing demand for software products is changing organizational structures in the software development industry. A reliable organizational structure enhances software quality, minimizes production costs, and enables timely delivery by adopting a strategic formation that manages and fosters interaction among team members (Massago, Leal, & Balancieri, 2018).

Agile Methodology: Scrum, Lean, and Kanban With GSD

The practitioner's efforts to overcome GSD challenges required agile methods in GSD settings. Alotaibi and Qureshi (2014) examined Scrum practices and temporal distance in GSD team coordination. Alsahil, Khan, and Alyahya (2017) reviewed resources on GSD challenges and examined several agile practices to mitigate problems, with a special focus on Scrum. Organizational leaders in GSD and modern researchers have continually experimented with various remedies to reduce GSD challenges. Agile practices such as regular communication and self-organization across remote locations could help minimize GSD challenges (Vallon, Estacio, Prikladnicki, & Grechenig, 2018). The combination of agile and GSD is a maturing field, with potential to improve software quality. Alvis, Goncealves, and Bax (2017) noted that Scrum helps drive process innovation and improvement. The combination of Scrum and Extreme Programming is a useful agile method, but defined empirical methods are needed to develop a better approach that aligns with GSD (Alvis et al., 2017; Vallon et al., 2018).

Practitioners and researchers applied and combined Lean, Lean Six Sigma, and Kanban with the GSD method to address certain software development concerns. Tanner and Dauane (2017) assert that lean thinking begins with value creation for the customer, while Kanban principles focus on value creation through visuals and work-in-progress management. Software companies choose development methodologies, which affect product quality and future markets. When applying Scrum principles, practitioners cannot add Lean principles of measurement and practice (Shahzeydi, Gandomani, & Sadeghi, 2018). Lean and Six Sigma are effective in IT-originated business processes when combined with Deming's system of profound knowledge and strong relationships, along with Agile, Lean, and Six Sigma (Krehbiel & Miller, 2018). Nataraja and Ramesh (2016) stated that the evolving trend in software development is continuous, with innovations in new methodologies, technologies, design tools, and emerging ideas.

Change Management

Challenges of organizational creativity compound complexities and dynamics in the GSD organizational environment. The frequency of organizational innovation and creativity is evident in a constantly changing GSD setting, as new technologies with unique requirements emerge. Changes and evolution in software development may significantly influence software quality (Jabangwe et al., 2016). Managing creativity to sustain innovation is one of the challenges for GSD organizations, given the changing market environment (Frank & Mengiste, 2014). Research on creativity in software organizations dates back to the 1990s. Frank and Mengiste (2014) applied a multi-adaptive method to obtain data from participants in Information Systems Development organizations and compare management perspectives and perceived challenges with findings in the literature. The key problem of software requirements engineering is change. RCM improves the requirements process. Alsanad, Chikh, and Mirza (2019) examined the use of ontology methodology in a GSD geographical setting to improve the requirements process by eliminating inconsistencies and incompleteness.

Global Software Development and Cloud Computing

The GSD and cloud computing are combined in an emerging strategy to overcome collaboration challenges and improve quality. The advantages include virtualization, reduced cost, scalability, infrastructure, and performance (Shanyour & Qusef, 2018). Service-Oriented Architecture (SOA) enables access to data applications. Shanyour and Qusef (2018) noted that SOA alleviates GSD coordination and knowledge-transfer problems. Ideas for using the cloud to address GSD challenges, such as collaboration, are expanding as software markets grow beyond national boundaries. Harshmi et al. (2011) confirmed that cloud computing features and elements can improve GSD strategy and products by leveraging knowledge gained from GSD and SOA. Cloud computing enhances agile global software development, despite agile, social, and technical concerns, because of the distributed environment (Haig-Smith & Tanner, 2016).

GSD Education and Training

Contemporary organizational GSD leaders proactively build awareness and train IT engineers on the complexities and socio-cultural challenges of GSD in dispersed, diverse teams. Chevers (2018) noted that company policy, requirements management, team member education, organizational climate, technical support, and end-user training influence quality and success. Aslam et al. (2017) investigated how GSD's competent workforces can increase skills acquisition for entrepreneurs and IT professionals in less-developed countries. Deming (1986) noted employee capacity building as one of the 14 principles of quality management theory. The global software market is rapidly reshaping through innovation, as businesses strive for best practices. Countries: India and China have become major players in the global markets in a short time because of the IT revolution. IT organizational leadership sustains GSD through system reviews, research, education, and to gain a competitive advantage (Talib, Saeed, Awais, & Hanif, 2017). As GSD evolves, workforce training and engagement depend on leadership strategies that improve employee outcomes and organizational investment gains.

Software Reuse Management

The availability of resources worldwide and ongoing innovation are among the reasons GSD is rapidly growing. scattered worldwide. Software organizations must meet increasing change and high stakeholder demand, despite project constraints. Organizations must build in quality and consider accessibility for every user at every stage of the software development life cycle, as larger audiences are becoming users of software systems (Baquero, Gil & Hernánde, 2018). The software reuse process is termed complex because of software legacy, despite the benefits of improved productivity and software quality (Paschali, Ampatzoglou, Bibi, Chatzigeorgiou, & Stamelos, 2017). GSD organizational leaders are viewed as developing and adopting better techniques to mitigate challenges and ensure high quality, productivity, and accessibility; reduce time and cost; simplify software testing; and enable software reusability. Kalia and Sood (2014) confirmed that standardization and organizational issues, such as infrastructure, legalities, and motivations, are among the challenges of component-based software reuse. To implement the reusability process during software development, practitioners use already developed software components. The quest to meet development needs led to software reuse. The component-based software reuse process involves selecting pre-tested components to reduce redesign and code development costs (Kalia & Sood, 2014).

Reuse frameworks help expedite the development process by reducing reliance on system components or existing markets, providing a quick response to new requirements. A. Khan, Khan, Amir, and Khan (2014) developed a component-based framework applying software architecture language identification, object-oriented design, and a software architecture orientation on a domain-specific and generic software product. These characteristics are common to component-based technologies and provide programmers with guidelines, prototypes, and principles (A. Khan et al., 2014). The demand for shorter development time and lower cost, with higher quality, encourages the use of component-based frameworks for software reusability. Existing software components must possess certain software architecture attributes, including functionality, behavior, and interfaces. A. Khan et al. (2014) used the SLR method to identify several component-based frameworks, tools, techniques, merits, and demerits. The derived component artifacts enter a repository for future use, while the second part of the framework analyzes the stored components

for reuse (A. Khan et al., 2014). Another aspect of the component-based framework includes software and component development, reusable domain analysis, reusable tagging, a reusable component repository, searching reusable artifacts, component integration, and validation (A. Khan et al., 2014). A. Khan et al. examined studies, compared different reusable techniques, and determined how to reduce cohesion between two independent components. Paschali et al. (2017) confirmed similarities with A. Khan et al. (2014) on component-associated software reusability. Paschali et al. (2017) identified existing literature and models on component-based software reuse and recommended a new model to improve software quality and reduce costs. To address the complexity of the reuse process, the proposed component-based software life cycle model consists of three phases: design, integration, and run time (Paschali et al., 2017). Some basic questions about component selection relate to defining the software component life cycle model. Reusable elements and progression, configuration problems require a solution (Paschali et al., 2017). Conventional reusable categories include code, design, and specification. Paschali et al. (2017) recommended that the study provide practitioners and researchers with a better understanding of software reusability.

Management of Software Security Perspectives

Software security is a necessary aspect of the SDLC, from requirements to final release (R.A. Khan & Khan, 2018). Software use is growing and more user-friendly in all aspects of life. Software applications were once common in business sectors, but now they support personal transactions and interactions, enhancing daily social networking. Internet-enabled software applications drive many processes, such as phone text messaging, email, video, telephone calls, online banking, and ATMs. Software is used extensively in many fields of endeavour, such as medical, education, telecommunication, agriculture, climate, finance, transportation, operations, industry, administration, military, research, and entertainment (Kalia & Sood, 2014). This high demand posed many challenges for meeting requirements and maintaining software product quality. Security, safety, data protection, personal privacy, and regulation became key requirements from the viewpoints of customers and end-users. Software security systems are internet-enabled computer software applications that are vulnerable to frequent malicious attacks (R.A. Khan & Khan, 2018). Software on a device must be designed to function and perform well under any form of attack. Customers expect software to solve problems, serve its purpose or improve the standard of living. Assurance assessment ensures that system design and development are free from defects and vulnerabilities (Rashidi & Hemanyati, 2018).

Software security is not an afterthought; it is considered at every stage of the SDLC process. M.U.A. Khan and Zulkernine (2009) agreed that software security starts at the beginning of the life cycle in the traditional software development method. R.A. Khan and Khan (2018) identified perspectives on software security in the literature, empirical studies, and implementation research by GSD organization vendors. The authors examined state-of-the-art software security practices in the SDLC. They developed the Software Security Assurance Model (SSAM) to determine GSD vendors' readiness to adopt software security in their environment (R.A. Khan & Khan, 2018). The methodology was a systematic mapping study that compared outcomes from the literature review and led to suggestions for future empirical studies of GSD vendors to determine their approaches to software security in practice (R.A. Khan & Khan, 2018).

Risk and Trust Mitigations

GSD Organizational leaders are aware of internal and external risks in the software development process. Risk management is defined as the continuous process of identifying and controlling potential risks that may affect software development objectives, especially in complex, uncertain environments (Khraiweh, 2013). Risk impacts software development in an organization: reputation, software product quality, and industry competitiveness (Tavares, Sanches da Silva, & Diniz de Souza, 2019). Software development risks are bound to increase because GSD involves additional steps and new requirements decisions that threaten the management process (Galli, 2018). The web-enabled platform enhances GSD collaboration through innovations, ideas, and product lines for customers (D.C. Chou & Chou, 2011). GSD managers should consider the team's dispersed nature, geographical distance, cultural differences, and organizational structure when managing risk (Galli, 2018). Risk identification and categorization help

reveal impacts and how to address them in a software development life cycle (Aslam et al., 2017). Effective, flexible, and robust risk management can prevent poor product quality and rework (Husin et al., 2019). GSD virtual team identified trust as a challenge for collaboration among team members. A team with a high level of trust can easily achieve high productivity and software quality. A team without trust may still be productive, but it requires extra effort from team members to monitor each other for compliance (Trainer & Remiles, 2018). Trainer and Remiles (2018) investigated how software tool support can leverage trust awareness among globally distributed and dispersed teams. The research involved a literature review of how developers form reliance on one another, as well as visualization strategies used by 28 students and 12 software professionals (Trainer & Remiles, 2018). Trainer and Remiles reported that team members who used visualization formed accurate, positive attributions and a certain level of trustworthiness toward teammates who worked remotely; the authors also noted that the study's measure of trustworthiness was the imitation, which could not be fully ascertained.

CRITICAL REVIEW

This section is a summary of reviewed literature as shown in “(See Table 1)”. *Analysis and Comparison of GSD Techniques*. Indicated in the analysis are current ideas, knowledge and techniques; themes, key features, framework, models, tools, and strategies in the course of the literature survey of this study. The literature further underscores the imperative role of leadership and management in GSD organizations. The analysis covered key methodologies, comparisons, and potential software development techniques in geographically distributed settings. The literature also identified additional GSD challenges due to the evolving, dynamic, and complex business environment, as stated in “(See Table 2)”. *GSD Challenges in Organizations*. The table presents an organized structure that highlights GSD challenges that organizational leaders may address and recognize as intended beneficial success factors. This study showed that GSD problems could be solved by applying appropriate policies, guidelines, best management practices and techniques. The benefits would be avoiding software project failures, achieving software product quality, and meeting customer expectations. See the tables in the Appendices.

FUTURE WORK

This study focused on literature from both client and vendor organizations. These critical reviews provided a common understanding of these organizations and valuable insights into GSD challenges and improvement techniques. The reviews showed that Leaders and practitioners employed and managed GSD in their specific environments. Future researchers may expand the study through broader investigation or exploration using qualitative or quantitative methods to better understand how organizations predict software product quality and meet customer expectations. Knowledge in management practice, standards, new advancing technologies, and software development techniques is evolving rapidly. Researchers could study how these new ideas and techniques, such as Artificial Intelligence (AI) frameworks, impact or benefit global software development.

CONCLUSION

Software development in geographically distributed locations is becoming the norm, replacing the traditional collocated approach. The reasons for this shift include available cheap labor, increased demand, and greater acceptance and adaptation. Organizational Leaders in GSD could address the prevailing problems by turning challenges into success factors. Leaders may employ recommended GSD techniques and build capacity to predict software product quality accurately and effectively. Leaders could also adopt GSD techniques, best management practices and standards in global software development to achieve competitive market advantage.

REFERENCES

- Agburu, J., Anza, N., & Iyortsuun, A.S. (2017). Effect of outsourcing strategies on the performance of small and medium scale enterprises (SMEs). *Journal of Global Entrepreneur Research*, 7, 26–27. doi:10.1186/s40497-017-0084-0
- Ahmed, Z., Mansor, Z., & Ahmad, K. (2017). An analysis of knowledge management challenges in agile global software development. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(4), 63–66. Retrieved from <https://journal.utem.edu.my/index.php>
- Akbar, M.A., Sang, J., Khan, A.A., Amin, E.F., Nasrullahi, S., Shafiq, M., . . . Xiang, H. (2018). Improving the quality of software development process by introducing a new methodology—AZ model. *IEEE Access*, 8, 4811–4823. doi:10.1109/ACCESS.2017.278798
- Akbar, R. (2019). Tailoring agile-based software development processes. *IEEE Access*, 7, 139852–139869. doi:10.1109/access.2019.2944122
- Akbar, R., Khan, A.R., & Adnan, K. (2020). Software development process evolution in Malaysian companies. *Advances in Intelligence System and Computing*, pp. 139–150. doi:10.1007/978-981-32-9949-8_10
- Ali, N., & Lai, R. (2017). A method of software requirements specification and validation for global software development. *Requirements Engineering*, 22, 191–214. doi:10.1007/s00766-015-0240-4
- Alotaibi, A.F., & Qureshi, M. (2014). Scrum and temporal distance-based global software development. *International Journal of Computer Network and Information Security*, 6, 48–53. doi:10.5815/ijcnis.2014.06.07
- Alrawashdeh, T.A., Muhairat, M., & Althunibat, A. (2013). Evaluating the quality of software in ERP systems using the ISO 9126 model. *International Journal of Ambient Systems and Applications*, 1(1), 1–9. Retrieved from <https://www.zuj.edu>
- Alsanad, A.A., Chikh, A., & Mirza, A. (2019). A domain ontology for software requirements change management in global software development environment. *IEEE Access*, 7, 49352–49362. doi:10.1109/ACCESS.2019.2909839
- Alvis, E., Goncalves, C.A., & Bax, M. (2017). Agile methods from the viewpoint of information. *Informação & Informação*, 178–210. doi:10.5433/1981-8920.2017v22n3p178
- Alzoubi, Y., Gill, A., & Al-Ani, A. (2016). Empirical studies of geographically distributed agile development communication challenges: A systematic review. *Information and Management*, 53(1), 22–37. doi:10.1016/j.im.2015.08.003
- Ammad, G., Janjua, U., Madni, M., Cheema, M., & Shahid, A.R. (2017). An empirical study to investigate the impact of communication issues in GSD in Pakistan's IT industry. *IEEE Access*, pp. 1–28. doi:10.1109/ACCESS.2019.2953008
- Anwer, S., Wen, L., Wang, Z., & Mahmood, S. (2019). Comparative analysis of requirement change management challenges between in-house and global software development: Findings of literature and industry survey. *IEEE Access*, 4, 1–39. doi:10.1109/ACCESS.2017
- Arumugam, A., & Kaliamourthy, B. (2016). Global software development: A design to measure risk of global practitioners. *International Conference on Computing, Analytics and Security Trends*, 1–4. doi:10.18293/SEKE2016-077
- Asiri, S., & Qureshi, M. (2014). Proposed GSD model to solve coordination and communication problems. *International Journal of Modern Education and Computer Science*, 11, 25–30. doi:10.5815/ijmecs.2014.11.03
- Aslam, W., Ijazi, F., Lali, M., & Mehmood, W. (2017). Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development. *Journal of Information Science and Engineering*, 33, 1481–1500. doi:10.6688/JISE.2017.33.6.6
- Awais, M.A. (2016). Requirements prioritization: Challenges and techniques for quality software development. *Advances in Computer Science: An International Journal*, 5(2). Retrieved from <http://www.ACSIJ.org>

- Babar, M., & Lescher, C. (2014). Global software engineering: Identifying challenges is important and providing solutions is even better. *Information and Software Technology*, 56(1), 1–5. doi:10.1016/j.infsof.2013.10.002
- Baquero, L., Gil, C., & Hernández, M. (2018). The usability and accessibility as quality factors of a secure web product. *International Journal of Applied Engineering Research*, 13, 16288–16294. Retrieved from <http://www.ripublication.com>
- Barney, S., Muhankumar, V., Chatzipetrou, P., Aurum, A., Wolin, C., & Angelis, S. (2013). Software quality across borders: Three case study on company internal alignment. *Information and Software Technology*, 56, 20–38. doi:10.1016/j.infsof.2013.06.004
- Bhatti, M.W., & Ahsan, A. (2016). Global software development: An exploratory study of challenges of globalization, HRM practices, and process improvement. *HRM Practices*, 10, 649–682. doi:10.1007/s11846-015-0171-y
- Bhatti, M.W., & Ahsan, A. (2017). Global monitoring and control: A process improvement framework for globally distributed software development teams. *Journal of Global Information Technology Management*, 20(1), 43–63. doi:10.1080/1097198X.2017.1280303
- Birgun, S., & Altan, Z. (2019). A managerial perspective for the software development process: Achieving software product quality by the theory of constraints. *IGI Global*, pp. 243–250. doi:10.4018/978-1-5225-7865-9.ch013
- Boehm, B.W. (2017). Software cost estimation meets diversity. *IEEE/ACM 39th International Conference on Software Engineering Companion*, 1–15. doi:10.1109/ICSE-C.2017.159
- Bourgroun, Z., Saber, M., & Bouchentouf, T. (2017). The reconciliation of external and internal quality attributes of ISO using the OO quality concepts. *4th International Conference on Automation, Control Engineering and Computer Science*, 19, 52–55. Retrieved from http://ipco-co.com/PET_Journal
- Calabrese, J., Munoz, R., Pasini, A., Espnda, S., Boracchia, M., & Pesado, P. (2018). Assistant for the evaluation of software product quality characteristics proposed by ISO/IEC 25010 based on GQM-defined metrics. *Argentine Congress of Computer Science*, 164–175. doi:10.1007/978-3-319-75214-3_16
- Carmel, E. (1999). *Global software teams: Collaborating across borders and time zones*. Englewood Cliffs, NJ: Prentice Hall.
- Carrillo de Gea, J.M., Nicolas, J., Fernandez Aleman, J.L., Toval, A., & Idris, A. (2016). Are the expected benefits of requirements reuse hampered by distance? An experiment. *SpringerPlus*, 5, 20–27. doi:10.1186/s40064-016-
- Chadli, S.Y., Idris, A., Ros, J.N., Fernadez-Aleman, J.L., Carillo, J.M., & Toval, A. (2016). Software project management tools in global software development: A systematic mapping study. *SpringerPlus*, 5, 1–3. doi:10.1186/s40064-016-3670-7
- Chever, D.A. (2018). A mixed method approach to investigate the antecedents of software quality and information systems success in Canadian software development firms. *Electronic Journal of Information Systems Evaluation*, 21(2), 109–130. Retrieved from <http://www.ejise.com>
- Chimagus, P., Selamat, A., Ibrahim, R., & Mahin, M.N. (2014). Systematic literature review of software requirement prioritization. *Information and Software Technology*, 56, 568–585. doi:10.1016/j.infsof.2014.02.001
- Chou, D.C., & Chou, A. (2011). Innovation outsourcing: Risks and quality issues. *Computer Standards & Interfaces*, 33, 350–356. doi:10.1016/j.csi.2010.10.001
- Colomo-Palacios, R., Soto-Acosta, P., García-Peñalvo, F., & García-Crespo, A. (2012). A study on the impact of global software development in packaged software release planning. *Journal of Universal Computer Science*, 18, 2646–2668. doi:10.3217/jucs-018-19-2646
- Cusick, J.J., Prasad, A., & Tepfenlart, W.M. (2008). Chapter 6 global software development: Origins, practices, and directions. *Advances in Computers*, 74, 201–269. doi:10.1016/S0065-2458(08)00606-2

- Deming, W.E. (1986). *Out of the crisis*. Cambridge, MA: Center for Advanced Engineering Study, Massachusetts Institute of Technology.
- Deming, W.E. (2013). A system must be managed. In J.N. Orsini (Ed.), *The essential Deming: Leadership principles from the father of quality* (1st ed., p. 151). New York, NY: McGraw-Hill.
- Denhere, N., Horne, T., & Van der Poll, A. (2015). Managing globally distributed software development projects using virtual teams: A Middle East case study. *ACM*, 1–10. doi:10.1145/2815782.2815786
- Denny, N., Mani, S., Nadella, S., Swaminatha, M., & Samadal, J. (2008). Hybrid offshoring: Composite personae and evolving collaboration technologies. *Information Resource Management Journal*, 21(1), 89–104. Retrieved from <http://www.igi-global.com>
- Djouab, R., & Bari, M. (2016). An ISO 9126 based quality model for the e-learning systems. *International Journal of Information and Education Technology*, 6, 370–375. doi:10.7763/IJiet.2016.V6.716
- Ebert, C. (2018). 50 years of software engineering: Progress and perils. *IEEE Software*, 35(5), 94–101. doi:10.1109/MS.2018.3571228
- Enabudoso, P. (2020). *Organization Global Software Development Challenges of Software Product Quality*. Walden University. Dissertations and Doctoral Studies. Retrieved from <https://scholarworks.waldenu.edu/dissertations/9354/>
- Enabudoso, P., & Schuttler, R. (2026). Global software development challenges of software product quality in organizations. *Journal of Management Policy and Practice*, 27(1), 66–80. doi:<https://doi.org/10.33423/2aathb25>
- Filho, M.S., Pinheiro, P.R., & Albuquerque, A. (2015). Task allocation approaches in distributed agile software development: A systematic review. *Advances in Intelligent Systems and Computing*, pp.1–10. doi:10.1007/978-3-319-18473-9_24
- Filho, M.S., Pinheiro, P.R., Albuquerque, A., & Rodrigues, J. (2018). Task allocation in distributed software development: A systematic literature review. *Wiley Hindawi Complexity*, pp. 1–13. doi:10.1155/2018/6071718
- Frank, U., & Mengiste, S.A. (2014). The challenges of creativity in software organizations. *Advances in Information and Communication Technology*, 429, 16–34. doi:10.1007/978-3-662-43459-8_2
- Galin, D. (2018). *Project progress control*. Wiley-IEEE Press, 617–634. doi:10.1002/9781119134527.app3
- Gheorghe, M. (2015). Software quality validation for web applications developed using geographically distributed human resources. *Informatica Economica*, 16(2), 101–111. doi:10.12948/issn14531305/19.2.2015.10
- Haig-Smith, T., & Tanner, M. (2016). Cloud computing as an enabler of agile global software development. *Issues in Informing Science and Information Technology*, 13, 121–144. Retrieved from <http://www.informingscience.org>
- Harshmi, S., Clerc, V., Razavian, M., Manteli, C., Tamburri, D.A., Lago, P., . . . Richardson, I. (2011). Using the cloud to facilitate global software development challenges. *International Conference on Global Software Engineering Workshops*, IEEE Computer Society, 70–77. doi:10.1109/ICGSE-W.2011.19
- Hidayati, A., Purwandari, B., Burdiardjo, E.K., & Solichah, I. (2018). Global software development and capability maturity model integration: A systematic literature review. *Third International Conference on Informatics and Computing*, 1–7. doi:10.1109/IAC.2018.8780489
- Horta, V., Stroele, V., Schettimo, V., Oliveira, J., David, J.M., & Araujo, A.P. (2019). Collaboration analysis in global software development. *IEEE 23rd International Conference on Computer Supported Cooperative Work in Design*, 28–32. doi:10.1109/CSCWD.2019.8791857
- Hossain, M. (2018). Challenge of software quality assurance and testing. *International Journal of Software Engineering and Computer Systems*, 4(1), 133–144. doi:10.15282/ijsecs.4.2018.11.0044

- Huisman, M., & Rubin, J. (2018). Software quality tools and techniques presented in fase'17. *International Journal on Software Tools for Technology Transfer*, 20, 611–61. doi:10.1007/s10009-018-0504-7
- Hussain, S., Farid, S., & Mumtaz, I. (2019). Is customer satisfaction enough for software quality? *International Journal of Computer Science and Software Engineering*, 8(2), 40–47. Retrieved from www.IJCSSE.org
- Huzita, E.H.M., Leal, G.C.L., Balancieri, R., Tait, T.F.C., Cardoza, E., Pentead, R.R.M., & Vivian, R.L. (2012). Knowledge and contextual information management in global software development: Challenges and perspectives. *2012 IEEE Seventh International Conference on Global Software Engineering Workshops*, 43–48. doi:10.1109/ICGSEW.2012.12
- Iftikhar, A., Alam, M., Musa, S., & Suud, M.M. (2018). Trust development in virtual teams to implement global software development (GSD): A structured approach to overcome communication barriers. *IEEE Explore*, 1–5. doi:10.1109/ICETSS.2017.8324169
- Indumini, U., & Vasanthapriyan, S. (2018). Knowledge management initiatives in agile software development: A literature review. *National Information Technology Conference*, 1–10. doi:10.1109/NITC.2018.8550066
- Innovation, Science, and Economic Development of Canada. (2018). *Canadian ICT sector profile: Automotive, transportation and digital technologies branch*. Retrieved from <https://www.ic.gc.ca>
- ISO/IEC 9126-1. (2001). *Software engineering—Product quality—Part 1: Quality model*. Geneva, Switzerland: International Organization for Standardization.
- Jabangwe, R., Borstler, J., & Peterson, K. (2014). Handover of managerial responsibilities in global software development: A case study of source code evolution and quality. *Software Quality Journal*, 23, 539–566. doi:10.1007/s11219-014-9247-1
- Jabangwe, R., Wohlin, C., Petersen, K., Smite, D., & Borstler, J. (2016). A method for investigating the quality of evolving object-oriented software using defects in global software development projects. *Journal of Software Evolution and Process*, 28, 22–64. doi:10.1002/smr.1788
- Jaffar, R., Hussain, A., & Chiad, W. (2019). A new model for study of quality attributes to components-based development approach. *Periodicals of Engineering and Natural Sciences*, 7, 1177–1185. Retrieved from <http://pen.ius.edu.ba>
- Jain, R., & Suman, U. (2015). A systematic literature review on global software development life cycle. *ACM SIGSOFT Software Engineering Notes*, 40(2), 1–14. doi:10.1145/2735399.2735408
- Jayatilleke, S., Lai, R., & Reed, K. (2018). A method of requirements change analysis. *Requirements Engineering*, 23, 493–508. doi:10.1007/s00766-017-0277-7
- Kalia, A., & Sood, S. (2014). Component based software reusability and framing standards for components to be reused. *International Journal of Computer Science and Information Technologies*, 5, 1459–1462. Retrieved from <http://www.ijcsit.com>
- Keshta, I.M. (2017). Software cost estimation approaches: A survey. *Journal of Software Engineering and Applications*, 10, 824–842. doi:10.4236/jsea.2017.1010046
- Khan, A., Khan, K., Amir, M., & Khan, M.N.A. (2014). A component-based framework for software reusability. *International Journal of Software Engineering and Its Applications*, 8(10), 13–24. doi:10.14257/ijseia.2014.8.10.02
- Khan, A.A., Basri, S., Dominic, P., & Amin, F. (2013). Communication risks and best practices in global software development during requirements change management: A systematic literature review protocol. *Research Journal of Applied Sciences, Engineering and Technology*, 5(9). doi:10.19026/rjaset.6.3554
- Khan, A.A., Keung, J., Naizi, M., Hussain, S., & Ahmad, A. (2017). Systematic literature review and empirical investigation of barriers to process improvement in global software development: Client–vendor perspective. *Information and Software Technology*, 87, 180–205. doi:10.1016/j.infsof.2017.03.006

- Khan, A.A., Keung, J.W., Amin, F.E., & Al-Wadud, M.A. (2017). Spiimm: Toward a model for software process improvement implementation and management in global software development. *IEEE Access*, 5, 2169–3536. doi:10.1109/ACCESS.2017.2728603
- Khan, A., Keung, J., Hussain, S., & Bennin, K. (2015). Effects of geographical, socio-cultural and temporal distances on communication in global software development during requirements change management: A pilot study. *The Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering*, 159–168. doi:10.5220/0005331401590168
- Khan, A.A., Shameem, M., Kumar, R.R., Hussain, S., & Yan, X. (2019). Fuzzy AHP based prioritization and taxonomy of software process improvement success factors in global software development. *Applied Software Computing*, 83. doi:10.1016/j.asoc.2019.105648
- Khan, K., Khan, A., Aamir, M., Zulfikar, S., & Khan, M.N.A. (2013). Quality assurance assessment in global software development. *World Applied Sciences Journal*, 24, 1449–1454. doi:10.5829/idosi.wasj.2013.24.11.13286
- Khan, M., & Khan, A.W. (2018). Virtual team management challenges mitigation model (VTMCMM). *International Conference on Computing, Mathematics and Engineering Technologies*, 2(2), 88–93. doi:10.1109/ICOMET.2018.8346328
- Khan, M.U.A., & Zulkernine, M. (2009). *A survey on requirements and design methods for secure software development*. Technical Report No. 2009-562. Queen's University School of Computing website. Retrieved from <https://research.cs.queensu.ca/TechReports/Reports/2009-562>
- Khan, R.A., & Khan, S.U. (2018). A preliminary structure of software security assurance model. *Proceedings of 13th International Conference on Global Software Engineering*, 132–135. doi:10.1145/3196369.3196385
- Khan, S.U., & Azeem, M.I. (2014). Intercultural challenges in offshore software development outsourcing relationships: An exploratory study using a systematic literature review. *IET Software*, 8(4), 161–173. doi:10.1049/iet-sen.2013.0012
- Khan, S.U., & Niazi, M. (2012). Critical challenges in offshore software development outsourcing: An empirical study. *International Conference on Software Engineering*, 1–9. doi:10.2316/P.2012.780-019
- Khraiweh, M. (2013). Project planning measures in CMMI. *International Journal of Software Engineering & Applications*, 4(2), 103–121. doi:10.5121/ijsea.2013.4208
- Krehbiel, T.C., & Miller, D.P. (2018). Should agile be part of your quality management system? *Journal of International Technology and Information Management*, 27(1), 1–28. Retrieved from <https://scholarworks.lib.csusb.edu/jitim>
- Kumar, D. (2017). Requirement paradigms to implement the software projects in agile development using analytical hierarchy process. *International Journal of Decision Support System Technology*, 9(3), 28–29. doi:10.4018/IJDSST.201707010
- Lee, S.J., Koo, R., Varuttamaseni, A., Yue, M., Li, M., Cho, J., & Gookkang, H. (2020). Optimization of software development life cycle quality for NPP safety software based on a risk-cost model. *Annals of Nuclear Energy*, 135. doi:10.1016/j.anucene.2019.106961
- Magableh, A.A., & AlSobe, M.R. (2018). Securing software development stages using aspect-orientation concepts. *International Journal of Software Engineering Applications*, 9(6), 57–71. doi:10.5121/ijsea.2018.9605
- Mahmood, S., Anwer, S., Ziazi, M., Alshayeb, M., & Richardson, I. (2017). Key factors that influence task allocation in global software development. *Information and Software Technology*, pp. 1–15. doi:10.1016/j.infsof.2017.06.009
- Marinho, M., Luna, A., & Beecham, S. (2018). Global software development: Practices for cultural differences. *Engineering and Computer Science*, pp. 1–6. doi:10.1007/978-3-030-03673-2_22
- Massago, Y., Leal, G., Balancieri, R., & Galdamez, E. (2018). Towards classifying sociocultural aspects in global software development. *Journal of Information Systems and Technology Management*, 15, 1–5. doi:10.4301/S1807-1775201815007

- Matthiesen, S., & Bjorn, P. (2015). Why replacing legacy systems is so hard in global software development: An information infrastructure perspective. *Collaboration in a Globalized World*, pp.876–890. doi:10.1145/2675133.2675232
- Miguel, A., Maurico, D., & Rodríguez, A. (2014). A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications*, 5(6), 31–53. doi:10.5121/ijsea.2014.5603
- Musa, K., & Alkhateeb, J. (2013). Quality model based on COTS quality attributes. *International Journal of Software Engineering and Application*, 4(1), 1–8. doi:10.5121/ijsea.2013.4101
- Nagappan, N., Murphy, B., & Basili, V. (2008). The influence of organizational structure on software quality: An empirical case study. *Proceedings of the 30th International Conference on Software Engineering*, 521–530. doi:10.1145/1368088.1368160
- Nataraja, P., & Ramesh, B. (2016). A review on lean thinking for software process improvement. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(11), 131–136. doi:10.17148/IJARCCCE.2016.51128
- Niazi, M., Mahmood, S., Alshayeb, M., & Hroub, A. (2014). Challenges of the existing tools used in global software development projects. *The Ninth International Conference on Software Engineering Advances*, 385–389. doi:10.1049/iet-sen.2014.0130
- Niazi, M., Mahmood, S., Alshayeb, M., Riaz, M.R., Faisal, K., Cerpa, N., ... Richardson, I. (2016). Challenges of project management in global software development: A client-vendor analysis. *Information and Software Technology*, 80, 1–19. doi:10.1016/j.infsof.2016.08.002
- Ouriques, R.A.D.B., Wnuk, K., Gorschek, T., & Svensson, R.B. (2019). Knowledge management strategies and processes in agile software development: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, pp. 1–24. doi:10.1142/S0218194019500153
- Paschali, M., Ampatzoglou, A., Bibi, S., Chatzigeorgiou, A., & Stamelos, I. (2017). Reusability of open-source software across domains: A case study. *Journal of Systems and Software*, 134, 211–227. doi:10.1016/j.jss.2017.09.009
- Polillo, R. (2011). Quality models for Web 2.0 sites: A methodological approach and a proposal. *Computer Science Bibliography*, pp. 251–261. doi:10.1007/978-3-642-27997-3
- Prikladnicki, R., Damian, D., & Audy, J. (2008). Pattern of evolution in the practice of distributed software development: Quantitative results from a systematic review. *International Conference on Evaluation and Assessment in Software Engineering*, 1–10. doi:10.14236/ewic/EASE2008.11
- Rajagopalan, S. (2014). Review of the myths on original software development model. *International Journal of Software Engineering & Applications*, 5(6), 103–111. doi:10.5121/ijsea.2014.5607
- Rashidi, H., & Hemanyati, M. (2018). Software quality models: A comprehensive review and analysis. *Journal of Electrical and Computer Engineering Innovations*, 6(1), 59–76. doi:10.22061/JECEI.2019.1076
- Ramacharan, S., & Rao, K. (2016). Scheduling based cost estimation model: An effective empirical approach for GSD project. *Thirteenth International Conference on Wireless and Optical Communications Networks*, 1–11. doi:10.1109/WOCN.2016.7759881
- Razzak, M.A., Bhuiyan, T., & Ahmed, R. (2014). Knowledge management in distributed agile software development projects. *IFIP International Workshop on Artificial Intelligence for Knowledge Management*, 1–27. doi:10.1007/978-3-319-28868-0_7
- Rehman, S., & Khan, S.U. (2014). Software quality attributes in global software development: Initial results. *Scientific International (Lahore)*, 26, 1415–1419. Retrieved from <http://www.sci-int.com>
- Rustambekovich, Y.N., Gulyamov, S.M., Usmanov, N.B., & Mirzaev, D.A. (2017). Challenging the ways to determine the faults in software: Technique based on associative interconnections. *Procedia Computer Science*, 120, 641–648. doi:10.1016/j.procs.2017.11.290
- Ryan, K.T. (2016). Software processes for a changing world. *Journal of Software Evolution and Process*, 28, 236–240. doi:10.1002/smr.1756

- Saleem, N., Mathrani, S., & Taskin, N. (2019). Understanding the different levels of challenges in global software development. *14th ACM/IEEE International Conference on Global Software Engineering*, 76–77. doi:10.1109/ICGSE.2019.00027
- Shahzeydi, M., Gandomani, T.J., & Sadeghi, R. (2018). Quality aspects of continuous delivery in practice. *International Journal of Advanced Computer Science and Applications*, 9, 2010–212. doi:10.14569/IJACSA.2018.090527
- Shameem, B., Chandra, B., Kumar, R.R., & Kumar, C. (2018). A systematic literature review to identify human-related challenges in globally distributed agile software development: Towards a hypothetical model for scaling agile methodologies. *International Conference on Computing Communication and Automation*, 1–7. doi:10.1109/CCAA.2018.877753
- Shameem, M., Kumar, C., & Chandra, B. (2017). Challenges of management in the operation of virtual software development teams: A systematic literature review. *International Conference on Advanced Computing and Communication Systems*, 6–7. doi:10.1109/ICACCS.2017.8014695
- Shanyour, B., & Qusef, A. (2018). Global software development and its impact on software quality. *IEEE Xplore*, 1–6. doi:10.1109/ISIICT.2018.8613294
- Sharma, S., Kaur, P., & Kaur, U. (2015). Communication understandability enhancement in GSD. *International Conference on Futuristic Trend in Computational Analysis and Knowledge Management (ABLAZE)*. Springer Berlin Heidelberg, 172–177. doi:10.1109/ABLAZE.2015.7154962
- Sugiyanto, S., Rochimah, S., & Sarworsri, S. (2016). The improvement of software quality model for academic websites based on multi-perspective approach. *Journal of Theoretical and Applied Information Technology*, 86, 464–471. Retrieved from <https://www.jatit.org>
- Suman, U., & Jain, R. (2015). A systematic literature review on global software development life cycle. *ACM SIGSOFT Software Engineering*, 40(2), 1–14. doi:10.1145/2735399.2735408
- Talib, R., Saeed, Y.M., Awais, M., & Hanif, K. (2017). Interactive systems regarding global software development and offshoring. *Current Science*, 112, 2134–2138. doi:10.18520/cs/v112/i10/2129-2134
- Tanner, M., & Dauane, M. (2017). The use of Kanban to alleviate collaboration and communication challenges of global software development. *Issues in Informing Science and Information Technology Education*, 14, 177–197. Retrieved from <http://www.informingscience.org/Publications/3716>
- Tavares, B., Sanches da Silva, C., & Diniz de Souza, A. (2019). Project to improve risk management in agile projects. *International Journal of Software Engineering and Knowledge Engineering*, 29, 381–399. doi:10.1142/S0218194019500165
- Tiwari, A., & Chakraborty, P. (2015). Software component quality characteristics model for component-based software engineering. *IEEE Xplore*, 47–51. doi:10.1109/CICT.2015.40
- Trainer, E.H., & Redmiles, D.F. (2018). Bridging the gap between awareness and trust in globally distributed software teams. *Journal of Systems and Software*, 144, 328–341. doi:10.1016/j.jss.2018.06.028
- Usman, A., & Khan, A.W. (2018). Software outsourcing quality challenges model systematic literature review (SLR) protocol. *University of Sindh Journal of Information and Communication Technology*, 2(4), 196–201. Retrieved from <http://sujo2.usindh.edu.pk/index.php/USJICT/>
- Usmani, N., Hassan, R., & Mahmood, W. (2017). Impediments to requirement engineering in distributed team. *International Journal of Information Engineering and Electronic Business*, 6, 10–18. doi:10.5815/ijieeb.2017.06.02
- Vallon, B., Estacio, B., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practice in global software development. *Information and Software Technology*, 96, 161–180. doi:10.1016/j.infsof.2017.12.004
- Waheed, S., Hamid, B., Jhanjh, N.Z., Humayun, M., & Malik, N.A. (2019). Improving knowledge sharing in distributed software development. *International Journal of Advanced Computer Science and Applications*, 10(6). doi:10.14569/IJACSA.2019.0100656

- Wang, M. (2016). The impact of cultural values on Chinese students in American higher education. *Qualitative Report*, 21, 611–628. Retrieved from <http://nsuworks.nova.edu/tqr/>
- Wickramarachchi, W., & Lai, R. (2016). Effort estimation in global software development: A systematic review. *Computer Science and Information Systems*, 14, 393–421. doi:10.2298/CSIS160229007W
- Yaseen, M., Ali, S., & Ullah, N. (2016). An improved framework for requirement implementation in the context of global software development: A systematic literature review protocol. *International Journal of Database Theory and Application*, 9(6), 161–170. doi:10.14257/ijcta.2016.9.6.16
- Yaseen, M., & Ali, Z. (2019). Success factors during requirements implementation in global software development: A systematic literature review. *International Journal of Computer Applications*, 182(48), 17–23. doi:10.5120/ijca2019918728
- Yi, J., Qi, D., Tan, S.H., & Roychoudhury, A. (2015). Software change contracts. *ACM Transactions on Software Engineering and Methodology*, 24(3), 1–43. doi:10.1145/2729973
- Yue, Y., Ahmed, I., Wang, Y., & Redmiles, D. (2019). Collaboration in global software development: An investigation on research trends and evolution. *International Conference on Global Software Engineering*, 1–30. doi:10.1109/ICGSE.2019.00028
- Zafar, A., Saif, S., Khan, M., Iqbi, J., Ahkunzada, A., Wdood, A., . . . Alamri, A. (2018). Taxonomy of factors causing integration failure during global software development. *IEEE Access*, 6, 22228–22239. doi:10.1109/ACCESS.2017.2782843

APPENDIX

TABLE 1
ANALYSIS AND COMPARISON OF GSD TECHNIQUES

Research Themes	Key Features	GSD Techniques
A review of software quality models for the evaluation of software products	The model is an acceptable technique for enhancing quality management in software products (Miguel et al., 2014).	Models and Frameworks in GSD
The reconciliation of external and internal quality attributes of ISO using the OO quality concepts	Bourgroun, Sabre, and Bouchentouf (2017) evaluated IT assessment challenges to improve software product quality and maintenance cost using Object-Oriented (OO) program model properties	
Toward a model for software process improvement implementation and management in global software development	It is imperative to investigate the effectiveness of the Software Process Improvement Implementation and Management Model (SPIIMM) due to the problem of software process improvement in software development (A.A Khan et al., 2017)	Process Improvement Management
Improving the quality of software development process by introducing a new methodology—AZ model	The modern trend of GSD includes implementing a software process model to improve software quality, set tasks, establish project duration, and ensure continuous process improvement (Akbar et al., 2018).	

A method of requirements change analysis	Prioritizing requirements in a GSD organization is a serious consideration in the SDLC to achieve software product quality (Jayatilleke, Lai, & Reed, 2018).	Requirement Management
Impediments to requirement engineering in a distributed team.	Usmani et al. (2017) implemented the requirement improvement model using the methodology of SLR and questionnaires to gather data from stakeholders on what challenges they faced while in the requirement process in a GSD environment	
Communication risks and best practices in global software development during requirements change management: A systematic literature review protocol	Software product quality issues reports are mostly when there is the engagement of outsourcing relationships, which can lead to low-quality products (Agburu, Anza, & Iyortsuun, 2017)	Integration Management (Coordination, Communication and Collaboration)
Taxonomy of factors causing integration failure during global software development	A good understanding of integration failure may help develop an appreciation of most integration challenges in the GSD setting (Zafar et al., 2018).	
A review of software quality models for the evaluation of software products	The model is an acceptable technique for enhancing quality management in software products (Miguel et al., 2014).	Models and Frameworks in GSD
The reconciliation of external and internal quality attributes of ISO using the OO quality concepts	Bourgroun, Sabre, and Bouchentouf (2017) evaluated IT assessment challenges to improve software product quality and maintenance cost using Object-Oriented (OO) program model properties	
Toward a model for software process improvement implementation and management in global software development	It is imperative to investigate the effectiveness of the Software Process Improvement Implementation and Management Model (SPIIMM) due to the problem of software process improvement in software development (A.A Khan et al., 2017)	Process Improvement Management
Improving the quality of software development process by introducing a new methodology—AZ model	The modern trend of GSD includes implementing a software process model to improve software quality, set tasks, establish project duration, and ensure continuous process improvement (Akbar et al., 2018).	
A method of requirements change analysis	Prioritizing requirements in a GSD organization is a serious consideration in the SDLC to achieve software product quality (Jayatilleke, Lai, & Reed, 2018).	Requirement Management

Impediments to requirement engineering in a distributed team.	Usmani et al. (2017) implemented the requirement improvement model using the methodology of SLR and questionnaires to gather data from stakeholders on what challenges they faced while in the requirement process in a GSD environment	Integration Management (Coordination, Communication and Collaboration)
Communication risks and best practices in global software development during requirements change management: A systematic literature review protocol	Software product quality issues reports are mostly when there is the engagement of outsourcing relationships, which can lead to low-quality products (Agburu, Anza, & Iyortsuun, 2017)	
Taxonomy of factors causing integration failure during global software development	A good understanding of integration failure may help develop an appreciation of most integration challenges in the GSD setting (Zafar et al., 2018).	
An empirical study to investigate the impact of communication issues in GSD	Managerial difficulties comprise 77% of GSD communication problems (Ammad, Janjua, Madni, Cheema, & Shahid, 2017)	Organizational Structure
Towards classifying sociocultural aspect in global software development	The reliable organizational structure will enhance quality of software, minimize cost of production, enable timely delivery, and help meet users' expectations by adopting a strategic arrangement that manages and fosters interaction among team members (Massago, Leal, & Balancieri, 2018).	
Scrum and temporal distance-based global software development	Alotaibi and Qureshi (2014) examined scrum practices on temporal distance-based on GSD team coordination challenges.	The Use of Agile Plus: Scrum, Lean, and Kanban with GSD
Agile development overcomes GSD challenges: A systematic literature review	Alsahil, Khan, and Alyahya (2017) reviewed the literature on GSD challenges and searched several agile practices on how to mitigate software development problems with a special focus on Scrum	
Systematic literature review on agile practice in global software development	The agile practices of regular communication and self-organization at various remote locations could help minimize GSD challenges (Vallon, Estacio, Prikladnicki, & Grechenig, 2018)	Change Management
A method for investigating the quality of evolving object-oriented software using defects in global software development projects	The frequency of organizational innovations and creativities is evident in a constant changing GSD setting by the unfolding new technologies with peculiar requirements. Software development change and evolution may have a significant influence on software quality (Jabangwe et al., 2016).	

The challenges of creativity in software organizations	Managing creativity to sustain innovations is one of the challenges of GSD organization, due to the continuously changing market environment (Frank & Mengiste, 2014).	
Proposed GSD model to solve coordination and communication problems	Despite the benefits of GSD, there are challenges and issues with the continuous progression of technologies, due to an evolving trend that is heterogeneous in nature (Shanyour & Qusef, 2018).	Identify Global Software Development Challenges
Challenges of management in the operation of virtual software development teams: A systematic literature review	These challenges could be in control, coordination, and communication, due to socio-cultural, geographical, and temporary distance (Asiri & Qureshi, 2014; S.U.Khan & Niazi, 2012).	
A mixed method approach to investigate the antecedents of software quality and information systems success in Canadian software development firms	Chevers (2018) concluded that organization climate, user training, education, requirement management and managerial technical support influence quality and success	GSD Education and Training
Interactive systems regarding global software development and offshoring.	GSD is sustained by IT organizational leadership through the performance of system reviews, research, education, and competitive advantage (Talib, Saeed, Awais, & Hanif, 2017).	
Component based software reusability and framing standards for components to be reused	Kalia and Sood (2014) affirmed that standardization and organizational issues, like infrastructure, legalities, and motivations are some of the challenges of component-based software reuse.	Software Reuse Management
Reusability of open-source software across domains: A case study	A different aspect of the component-based framework includes software development, component development, reusable domain analysis, reusable tagging, reusable component repository, search reusable artifacts, component integration, and system validation (A.Khan et al., 2014).	
A preliminary structure of software security assurance model	Software security is a vital aspect of the SDLC from the requirement to final disposal in GSD (R.A. Khan & Khan, 2018).	Software Security Management
Software quality models: A comprehensive review and analysis	Software assurance assessment is ensuring the system design and building is free from defects and any kind of vulnerabilities (Rashidi & Hemanyati, 2018).	Risk and Trust Mitigations
Project planning measures in CMMI	Risk management is a continuous process that captures and controls potential risks that may affect the objectives of the software development information system, especially in a complex environment with uncertainties (Khraiwesh, 2013).	

Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development	Risk identification and categorization helped to reveal the impacts and how to address it in a software development life cycle (Aslam et al., 2017).
Bridging the Gap Between Awareness and Trust in Globally Distributed Software Teams	A team without trust may still have high productivity, but with extra-effort of team members monitoring each other for compliance (Trainer & Remiles, 2018).

TABLE 2
GSD CHALLENGES IN ORGANIZATION

GSD Challenges In Literature	Study Citations	GSD Leader/Manager Mitigations Requirement (Yes/No)	Target Success Factors
Human (People)	(Shameem, Chandra, Kumar, & Kumar, 2018),	Yes	√
Poor quality	(Jaffar, Hussain, & Chiad, 2019),	Yes	√
Technical	(Suman & Jain, 2015),	Yes	√
Political	(Suman & Jain, 2015),	Yes	√
Social-Cultural	(Suman & Jain, 2015),	Yes	√
Distance (Geographical or Temporary)	(Carrillo de Gea, Nicolas, Fernandez-Aleman, Toval, & Idris, 2016),	Yes	√
Speed	(Carrillo de Gea, Nicolas, Fernandez-Aleman, Toval, & Idris, 2016), (Marinho, Luna, & Beecham, 2018).	Yes	√
Requirement	(Marinho, Luna, & Beecham, 2018).	Yes	√
Standardization	(Jain & Suman, 2015),	Yes	√
Stakeholder Management	(Colomo-Palacios, Soto-Acosta, García-Peñalvo & García-Crespo, 2012),	Yes	√
Trust	(Trainer & Remiles, 2018),	Yes	√
Risk	(Aslam, Ijaz, Lali, & Mehmood, 2017),	Yes	√
Replacing legacy systems	(Matthiesen & Bjorn, 2015).	Yes	√
Language difference	(Denhere et al., 2015).	Yes	√
Communication	(Babar, & Lescher, 2014; Yue, Ahmed, Wang, & Redmiles, 2019).	Yes	√
Coordination	(Babar, & Lescher, 2014; Yue, Ahmed, Wang, & Redmiles, 2019).	Yes	√
Collaboration	(Babar, & Lescher, 2014; Yue, Ahmed, Wang, & Redmiles, 2019).	Yes	√
Change management		Yes	√

knowledge sharing	Huisman and Rubin (2018).	Yes	√
New technologies	(Shanyour & Qusef, 2018), (Yassen & Ali, 2019).	Yes	√
Geographical difference	(Denhere et al., 2015).	Yes	√
Time-zones			
Policies	(Denhere et al., 2015).	Yes	√
Governance	(Denhere et al., 2015).	Yes	√
Religion	(Denhere et al., 2015).	Yes	√
Work Ethics	(Denhere et al., 2015).	Yes	√
Tools and compatibility	(Denhere et al., 2015).	Yes	√
Global security	(Denhere et al., 2015).	Yes	