# A Review of Human Factors Research in Distributed Software Engineering

**Austin D. T. FitzGerald, University of Wisconsin - Platteville**

## Abstract

As recently spotlighted by the COVID-19 pandemic, shifts towards distributed working environments result in dichotomous experiences for software engineers. The advent of a pandemic is a factor of its own. However, past research in distributed software engineering (DSE) contains observations of the personal, interpersonal, and organizational human factors that result in such dichotomies. This paper details the current state of studies regarding DSE and resultant impacts on software quality. Areas of software engineering which involve sociotechnical human factors are largely understudied. Only fragments of information connect studies involving DSE. Nevertheless, existing literature contains patterns into the effects of distribution on communication, coordination, and cooperation among software engineers and analogous careers. Their experiences intertwine with these factors, along with awareness and trust. Empirical studies reveal the use of diverse software quality metrics, leading to a set of conclusions comparable among studies. Still, the effects of distribution on software quality are unclear. Before bodies of knowledge can contain accepted DSE practices, further collaboration is required in research fields involving human factors in software engineering.

## 1) Introduction

The advent of the COVID-19 pandemic forced software engineers across the world to shift into remote work, and their dichotomous experiences are already making their way into publications [23, 49]. This shift's scale and speed was unlike any before, but literature involving the analysis and review of distributed software engineering (DSE) is not as new. The evolution of technology has dictated globalization of markets, especially in software engineering. As a result, research regarding software engineering in a distributed context has increased. The purpose of this literature review is to synthesize concepts provided in recent research and provide an overview of the human factors involved in DSE. The process of gathering and exploring literature was carried out using the following parameters.

1. Studies published after the year 2000 were considered for review. This choice of the year was intended to limit the review's scope to the most recent of research. Any work in this subject area before the year 2000 should be considered of no less importance. Those authors are thanked for their work in a field that was mostly barren at that time.
2. Only studies published in the English language were considered for review; this was not intended to limit the scope to research conducted in primarily English-speaking regions.

Instead, it was to appease this author's monolingualism. Literature from globally dispersed authors was purposefully pursued.

3. Systematic reviews and mapping studies were leveraged to find other research but were not used as primary sources unless they made unique contributions.

Within these parameters, the following databases and catalogs were utilized: ACM Digital Library, arXiv, Gale, Google Scholar, IEEE Xplore, ResearchGate, ScienceDirect, Search@UW, SpringerLink, and Wiley Online Library. Searches included various combinations of the following keywords: (dislocated, dispersed, distributed, global, nearshore, offshore, offsite, virtual, work from home) and (quality, security, software) and (development, engineering, organizations, teams) and (effect, failure, impact, improvement) and (aspects, awareness, barriers, capabilities, communication, coordination, culture, factors, happiness, human, impediments, information sharing, obstacles, sociotechnical) and (case study, empirical, review, taxonomy). Zotero was used as a reference-management tool.

## 2) Distributed Software Engineering

Distributed software engineering (DSE), also referred to as distributed software development (DSD), is a term that describes many related organizational structures. Unfortunately, this growing field has no unified terminology or taxonomy. Many features described as part of DSE or DSD overlap across different papers, but authors rarely define the terms used [53]. The lack of standardization leaves research fragmented, and few authors have made a notable effort into untangling the web of terminology. Two papers to date, written by Gumm [25] in 2006 and Šmite et al. [53] in 2014, contain descriptions that best fit those used by other authors. Through literature review and expert surveys, both papers describe DSE as a facet of geographic location and relationships involving inter-organizational and intra-organizational structures [25, 53]. Furthermore, the latter study is the only one to map an empirically based DSE taxonomy (Figure 1). This taxonomy is useful in classifying the organizational structures described in existing studies, it cannot delineate the descriptions of DSE in all papers.

For example, the taxonomy created by Šmite et al. uses geographic distance as a level of DSE, further defining "far" and "near" as describing classes [53]. They make no distinction of global distribution. In contrast, other authors define global software engineering/development (GSE/GSD) as a distinct subtype of DSE determined by international distribution [25, 47]. Even so, these and other authors have used the terms and descriptions of DSE/DSD/GSE/GSD interchangeably. For the sake of continuity, this literature review will continue to use the term DSE and note that GSE/GSD is an artifact of international geographic distance. Another example of disagreement involves the inclusion of culture as a distinct factor. The distribution of software engineering has increased cultural diversity within organizations. Many of the papers included in this literature review describe challenges in DSE involving cultural differences such as language, politics, religion, work ethics, and more. Thus, there are arguments that culture may not only be a factor of geographic location but also a distinct dimension of DSE [2, 8, 11, 15, 48]. The Šmite et al. [53] paper states that the impact of culture on DSE is vitally important to consider but does not explain the lack of its inclusion as a dimension separate from geographic location.

Furthermore, common terms such as dispersed, distributed, remote, and virtual software engineering teams are examples of names for low-level organizational structures that are not defined in existing taxonomies for DSE. In one paper, a distributed team is defined as a group of geographically distributed individuals who work on the same executable for a project [7]. Contrarily, that description of a distributed team is more akin to that of a virtual team in a different paper, whereby virtual team members work “jointly on the same tasks” and distributed team members do not [31]. However, in other papers, the terms are used interchangeably as a general description of teams that are project-specific and contain any members who are geographically distributed [4, 25, 42, 43, 45, 50].

These differences in basic terminology highlight the fragmentation of knowledge in the field of distributed software engineering. A more widely accepted taxonomy, as stated by Šmite et al., “would potentially result in improvement of our understanding of individual strategies” [53].

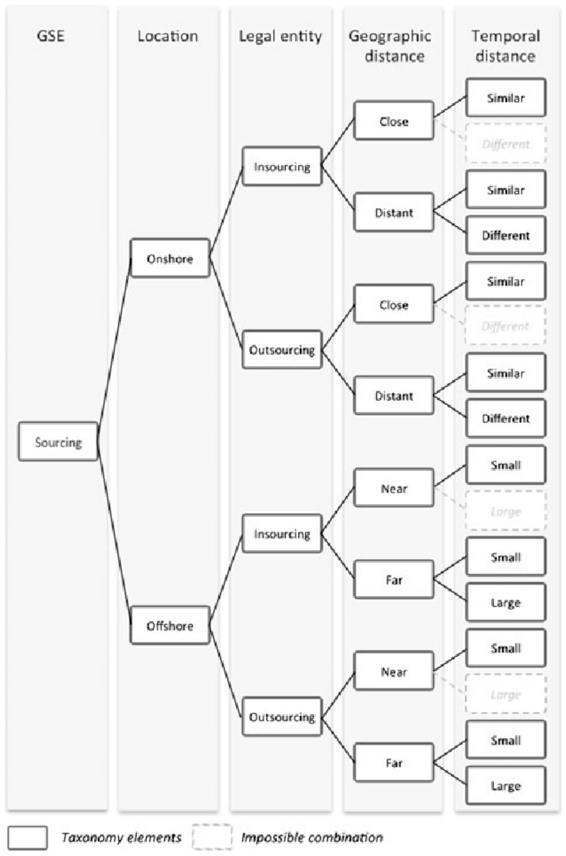

Figure 1: A taxonomy of DSE [53]

## 3) Trust, Awareness, and the 3C Model

A distributed environment directly impacts conventional software engineering practices at each phase of the development lifecycle [37]. These impacts can be described through the lenses of communication, coordination, and cooperation [52]. Titled the “Three C’s” or the “3C Model”, these three terms are used widely throughout literature involving DSE. However, as is similar to general DSE terminology, most papers do not contain definitions or references for the terms. In the following paragraphs, the definitions of each term in the 3C Model are adapted from multiple authors' work. One paper explicitly provides definitions in the context of software engineering [36]. Another paper describes the model in the context of general collaborative systems, shown in Figure 2 [24]. The last paper provides a diagram of the author's perceived barriers in DSE, shown in Figure 3 [11]. It is worth noting that some authors use the term collaboration in place of cooperation. Trust and awareness are two of the most explored themes in human factors literature [39]. These interpersonal factors are critically interrelated with the 3C Model [19, 44, 55].

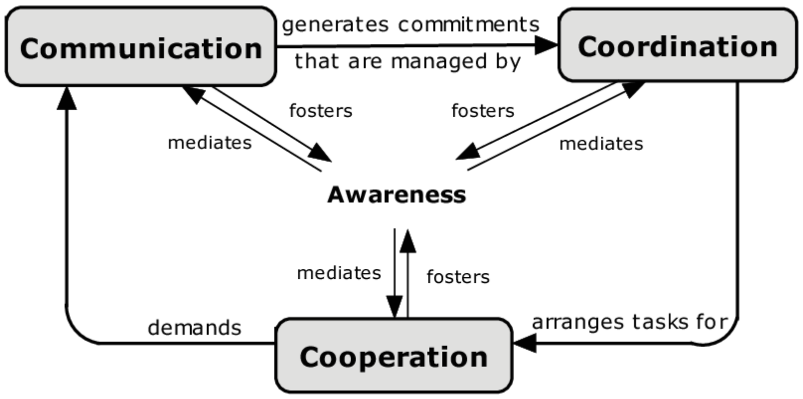

Figure 2: The 3C Model for general collaborative systems [24]

Researchers usually approach trust between humans and machines. However, when designing sociotechnical systems, interpersonal trust is a critical factor. Frameworks that model the trust formation process allow for a better ability to develop and maintain collaborative systems [39]. An article titled “Bridging the Gap Between Awareness and Trust in Globally Distributed Software Teams” describes interpersonal trust in DSE as the positive or negative expectations people have about each other's behavior [55]. Attributions, ways people reason on the cause of events, form these expectations and make up a person's perceived trustworthiness.

A fundamental attribution error occurs when a person makes an attribution based on someone's characteristics when they should have instead based it on the event's circumstances. As noted by one expert in Globally Distributed Software Development [10], fundamental attribution errors are more common in DSE due to the lack of social presence. Without natural social context, trust must be “actively facilitated, fostered and developed” instead of formed through natural awareness [11]. The author also presents findings from four independent case studies involving onshore and offshore distribution with varying geographic distance. Each of these studies exposes a “them and us” culture between team members in different locations. Failures in project management resulted in clear examples of anti-cooperation and a lack of knowledge sharing between remote colleagues.

Awareness is the bridge between trust and the 3C Model. In order for members of software engineering processes to effectively collaborate, they must be aware of both formal and informal aspects involving each other. In “A review of awareness in distributed collaborative software engineering”, the authors define the dimensions of awareness as follows [44].

- Informal awareness involves knowing what group members are doing.
- Group-structural awareness involves knowing the roles of group members.
- Social awareness involves knowing the state of attention, interests, and emotions of group members.
- Workspace awareness involves knowing the resources related to coordination.

The authors also describe a fifth dimension, context awareness. The aspects of this dimension crosscut the others. If each of the four basic dimensions is a pool of knowledge that is developed and maintained, then context awareness is the information about their interrelatedness and state over time. In other words, awareness requires context, which is a form of awareness in itself. This context allows people to make more accurate attributions of trust [55]. Some of awareness development is natural; a person will learn more about their colleagues with time. Other parts need to be actively maintained; it requires a project timeline to keep everyone on the same page. Unfortunately, developing and maintain awareness is problematic in DSE [16, 33]. Processes involving communication, coordination, and cooperation require changes in order to establish means of awareness development comparable to traditional SE.

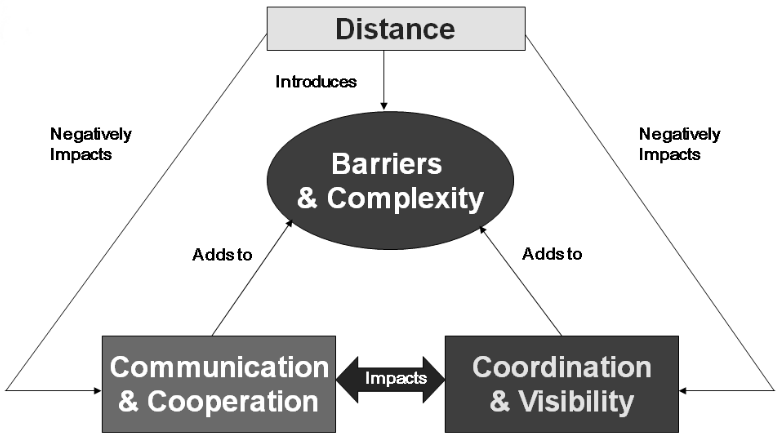

Figure 3: The 3C Model as used to describe DSE barriers [11]

Communication is the exchange of information via any media between people [24, 36]. It is possibly the single most referenced factor related to DSE challenges [7]. Traditional face-to-face meetings are far less common, so communication requires some technological medium. One notable paper describes the need for adequate communication mechanisms in DSE in terms of media richness [4]. Specifically, reducing ambiguity in communication requires rich media such as video conferencing. Likewise, another study found that the loss of face-to-face communication introduced significant project delays due to inadequate tools available for rich communication [27]. Even with rich media, an inability to interpret body language or eye contact can lead to a lack of awareness and less creative discussion [33]. Likewise, language differences may relate to unbalanced communication in DSE [54]. One paper states that the richness of media can never overcome the hindrances on communication caused by distance and cultural differences [43]. However, this claim is disputed in recent years. In a direct refutation, one set of authors state that "distance does not have as strong of an effect on distributed communication delay and task completion as we have seen in past research" [41]. Nevertheless, distribution can positively impact communication in some environments, as the freedom to choose the richness of media allows for more efficient communication depending on the circumstance [35].

Coordination is the organization of people and tasks [24, 36]. The success of software engineering in a distributed environment relies on successful coordination [26]. There can be many of individuals or teams that are owners of certain parts of a distributed software project. For software parts to eventually compose an entire product, individuals and teams must coordinate. They must share schedules, allocate responsibilities, and maintain dependencies of tasks onto individuals and teams [19]. Therefore, coordination leverages communication and cooperation by engaging people in meetings to create schedules, responsibilities, and tasks [24, 32, 38]. Geographic and temporal dispersion result in increased complexity and cost of coordination [18]. People engaging in DSE may find themselves frequently missing release dates due to the complexity of cross-team coordination created by distance [19]. Some researchers have observed that organizational support for defined coordination structures should be established early in a project's timeline. Software architects, who begin to work in the earliest stages of development, have a significant role in shaping the mechanisms for coordination [26]. Likewise, both developers and stakeholders must define involved parties, expectations for coordination, scheduling, and other mechanisms early in a project [11, 54]. Thus, coordination relies heavily on individual and team awareness.

Cooperation is the process of people working together on tasks in a shared environment, it describes actions of collective work [24, 36]. Instances of cooperation are directly supported by communication and coordination, thus distribution also negatively impacts the ability for people to effectively cooperate [11, 37]. There are plenty of existing tools to leverage in distributed cooperation, and some studies have claimed more importance of consistency in choosing tools than the specific tools themselves [7, 47, 48, 54]. In practice, cooperation is dependent mainly on the engineering approaches used. One such approach, Agile, allows organizations to maintain specific human roles, timelines, and meetings for SE teams [14]. Multiple studies have shown success in DSE after carefully planned implementations of SE methodologies such as Agile and eXtreme Programming, as well as techniques like pair programming [14, 36, 48]. Software engineers must be motivated to effectively cooperate with each other in a distributed environment, and much of this responsibility lies with management [51]. Teams which consistently demonstrate progress towards goals are those which are possess a cohesive and honest atmosphere [8]. As stated in one engineering management textbook, "coordination boils down to two conditions: 1) that people and units know what they are to do and 2) when they are to do it" [5]. However, software engineers cannot rely solely on their individual cognition, they must be able to consult with others to solve problems and work towards large goals [8]. Cooperation requires trust between coworkers and units, no matter where they are located.

## 4) Impacts of DSE on Software Quality

Software engineering is a sociotechnical activity entrenched in human-centered design. In this context, human-centered design is a sustainable approach in SE that addresses ergonomics regarding users, developers, and other stakeholders [28]. As a human-centered activity, each phase of SE is greatly affected by each parties' feelings, attitudes, and behaviors [1, 34].

As discussed in the preceding section, researchers use the concepts of trust, awareness, and the 3C Model to describe impacts of distribution on human-centered design. The makeup of DSE literature is largely formulative. Researchers provide creative explorations of challenges in DSE through the use of interviews, case studies, and interviews. The conclusions that researchers provide aim to alleviate sociotechnical concerns introduced by distribution in software engineering processes. Even with consideration of the immaturity of the DSE field, researchers often state similar lessons and suggestions. An unordered list of synthesized conclusions is as follows.

1. A balanced and consistent mix of synchronous and asynchronous communication, with the freedom to choose from the newest tools, mitigates many social concerns with distributed communication [7, 17, 50]. It may be helpful to explicitly define a communication model and train people on it [2, 54].
2. Information sharing, or knowledge sharing, is conducive to the process of building contextual awareness [26, 47]. Multiple authors describe overcoming the struggles of coordination in DSE by utilizing daily meetings explicitly designed around the concept of information sharing [19, 33, 35, 37, 54]. There must be extensive and transparent documentation of any and all decision-making, especially information that could be implicitly shared in a face-to-face setting [35, 50].
3. Cooperative processes must be explicitly defined during the planning stages for project management [38, 47]. Project managers are responsible for ensuring that all objectives are understood by personnel [11]. Some authors describe a need for small team size [2, 30, 54], but not all agree that it is of significant concern [17, 40].
4. Managers should organize workshops, informal social meetings, and establish a formal trust building process [33, 37, 51, 55]. This also enables further awareness of cultural profiles on and between teams of different sites [7, 37, 51].

These conclusions are not unique to the literature reviewed. This is evident when reviewing *The Guide to the Software Engineering Body of Knowledge (SWEBOK)*, a book which is listed as an international standard and is a foundation for deciding university accreditation criteria [20, 29]. Each of the topics covered in *SWEBOK* are accompanied by references that are noted to be complete, sufficient, consistent, credible, current, and succinct [8]. *SWEBOK* has two notable sections that detail accepted elements of professional practice for software engineers. The sections, titled “Group Dynamics and Psychology” and “Communication Skills”, contain conclusions that are remarkably similar to those synthesized above [8]. Interestingly, only few of these topics contain descriptions of any factors related to distribution. It is possible that the synthesized conclusions identified above should be considered areas of extra focus. Researchers have largely identified those conclusions, instead of other areas described in *SWEBOK*, as success factors for DSE. Nevertheless, the origin of their conclusions is not usually empirical. There are very few empirical studies that measure the impacts of distribution on software quality; of those studies, results can be conflicting.

Measurements of software defects are a common way to investigate the impact of DSE on quality. Jabangwe et al. [30] detailed a study of two commercial products developed throughout the same company, but each spread between Sweden and Russia. In the Šmite et al. [53] taxonomy, this would be considered offshore insourcing. Each of these projects were also undergoing a project management shift. The authors state that such shifts are usually linked to a decrease in quality. In this study, software quality is evaluated using metrics from source code (internal quality) and reported defects (external quality). Source code metrics included size measures such as Lines of Code and complexity measures like Average McCabe Cyclomatic Complexity. The authors have found these measures to be reasonable indications of quality in the past. Defect data came from post-release customer reported issues that were a result of a deficiency in source code. Extensive analysis was performed throughout multiple release cycles. Defect data was studied using descriptive statistics and visualizations over time. Source code measures were analyzed with heat maps and moving-range charts. Jabangwe et al. [30] found no observable impact of distribution on quality, attributing a list of success factors.

Bird et al. [7] conducted a similar study in 2009. These authors studied the source code base for Windows Vista, which included contributions from 2,757 developers across Asia, Europe, and North America. The authors described various geographic location classifications of building, cafeteria, campus, locality, continent, and world. Engineers worked in a distributed context in each of these categories, each of which introduced different elements of location, geographic distance, temporal distance, and cultural barriers. Thus, the entire study cannot be swiftly mapped to the Šmite et al. [53] taxonomy. However, the project is solely involved with insourcing. The source code for individual executables and libraries was mapped to commit logs and a geographic classification. Like the methodology used in Jabangwe et al. [30], external and internal code quality metrics were collected. The researchers studied post-release failures, code size and complexity, code churn, test coverage, and dependencies between binaries. They found that there was no significant evidence of adverse effects to quality from distribution. The paper contains a conclusion of positive sociotechnical practices provided from discussions with both management and senior employees.

Cataldo and Nambiar [13] conducted a study of the impacts of distribution on software quality using data from an embedded systems company. The data spanned 189 projects that were distributed across Europe, India, and Japan. Regarding the Šmite et al. taxonomy, this organizational structure resembles insourcing, but further distance elements vary. The authors measured software quality as the number of defects reported during integration and system testing for each project. Measures of external software quality were not used. An impressive set of independent variables were unlike those used in other papers. Two indexes, spatial distance and temporal distance, were used to compare two locations, the number of people at each, and the number of people in the project total. Also, the authors conceptualized configurational dispersion as measures of people-based dispersion and modification-request-based dispersion. Finally, control factors related to code size and churn and architectural component modification were captured. After examining and discussing the results of multiple logistic regression models, Cataldo and Nambiar [13] concluded that multiple dimensions of DSE have independent impact on quality. However, they did not conclude that spatial distribution negatively impacts software quality on a broad level.

Multiple other studies extend conclusions involving negative impacts on software quality due to distribution. Even two authors from an aforementioned study [7], Bird and Nagappan, have found contradictory results in different software projects. In evaluating pre and post-release defects in the Eclipse and Firefox codebases, these authors found that “all measures of geographic and organizational distribution increase failures, but the effects are not consistent across releases" [6]. Likewise, Cataldo and Nambiar concluded in a later study that higher levels of temporal dispersion in a team negatively affected software quality [12]. The effects of team size are also massively contested.

One especially interesting study involves a deeper dive into why these contradictory findings may occur. Motivated by a "quest to find empirical evidence of the effect of distance on software artifacts," the authors explore the idea of aggregation bias in a large case study [40]. Aggregation bias occurs when evaluating a hypothesis on software artifacts contradicts that on the aggregate of those artifacts. Previous empirical evidence of this type of bias is reported in multiple other studies [40]. The case study involved data from the IBM Rational Jazz project over fourteen months. The project was distributed among sixteen sites in the United States, Canada, and Europe. The authors chose to evaluate two software quality constructs, the time for a work item resolution and the defect count. Data was collected based on these quality measures for software artifacts and their aggregate components. The paper details statistical analyses: distribution comparisons, controlled comparisons, and multivariate analysis. Nguyen et al. [40] conclude that the effects of distribution can be observed at the artifact level, but not at the component level. The authors note that this confirms that aggregation bias may explain the contradictory findings, but extensive future work is needed. Unfortunately, this study has seemingly gone unnoticed by other researchers.

## 5) Conclusion

The impacts of distribution on software quality are unclear. At most, the synthesis of current literature results in the conclusion that software quality is dependent on the measures used, elements of distribution, and the extent of mitigating factors. Even those success factors from studies on the effects of distribution on software quality are not remarkably different from accepted practices already identified in *SWEBOK*. Despite the critical role of human factors, researchers have only just begun to unravel their impacts in software engineering [9]. Longstanding overrepresentation of technical aspects has obstructed the advancement of practices in both the classroom and the workplace. In what one author considers to be the only comprehensive review of human factors in software engineering [3], Pirzadeh [46] concluded that primary study researchers have acutely overlooked human factors in software engineering, specifically human factors at the interpersonal level. Even the few studies that have considered personal factors of developers reportedly used outdated metrics [21].

Furthermore, even though academic societies such as IEEE CS and ACM call for university curricula that dives deeper into human factors, progress is not apparent [9]. Capretz, an established researcher and educator, states that "[T]his won't change until we realize that the human element is pivotal to software engineering and that it's worthwhile studying and teaching this so-called soft subject. However, few courses in any computer science or software engineering curricula even mention it" [9]. Fortunately, the number of papers dealing with human factors in SE is on an upward trend in the past twenty years [3]. Likewise, even considering the fragmentation of DSE research, Šmite et al. [53] provide a positive outlook for future research and practice. There is work to be done, but research is headed in the right direction.

## References

[1] Acuna, S. T., Juristo, N., & Moreno, A. M. (2006). Emphasizing human capabilities in software development. *IEEE Software*, *23*(2), 94–101. https://doi.org/10.1109/MS.2006.47

[2] Al-Ani, B., & Edwards, H. K. (2008). A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams. *2008 IEEE International Conference on Global Software Engineering*, 35–44. https://doi.org/10.1109/ICGSE.2008.9

[3] Amrit, C., Daneva, M., & Damian, D. (2014). Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. *Information and Software Technology*, *56*(12), 1537–1542. https://doi.org/10.1016/j.infsof.2014.07.006

[4] Andres, H. (2002). A comparison of face-to-face and virtual software development teams. *Team Performance Management*, *8*, 39–48. https://doi.org/10.1108/13527590210425077

[5] Babcock, D. L., Morse, L. C., & Schell, W. J. (2020). *Managing engineering and technology* (Seventh edition). Pearson Education, Inc.

[6] Bird, C., & Nagappan, N. (2012). Who? Where? What? Examining distributed development in two large open source projects. *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, 237–246. https://doi.org/10.1109/MSR.2012.6224286

[7] Bird, C., Nagappan, N., Devanbu, P., Gall, H., & Murphy, B. (2009). Does distributed development affect software quality? An empirical case study of Windows Vista. *2009 IEEE 31st International Conference on Software Engineering*, 518–528. https://doi.org/10.1109/ICSE.2009.5070550

[8] Bourque, P., Fairley, R. E., & IEEE Computer Society. (2014). *Guide to the software engineering body of knowledge*.

[9] Capretz, L. F. (2014). Bringing the Human Factor to Software Engineering. *IEEE Software*, *31*(2), 104–104. https://doi.org/10.1109/MS.2014.30

[10] Casey, V. (n.d.). *Dr. Val Casey | UL - University of Limerick*. Retrieved January 16, 2021, from https://www.ul.ie/research/dr-val-casey

[11] Casey, V. (2010). Developing Trust In Virtual Software Development Teams. *Journal of Theoretical and Applied Electronic Commerce Research*, *5*(2), 41–58. https://doi.org/10.4067/S0718-18762010000200004

[12] Cataldo, M., & Nambiar, S. (2009). On the relationship between process maturity and geographic distribution: An empirical analysis of their impact on software quality. *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, 101–110. https://doi.org/10.1145/1595696.1595714

[13] Cataldo, M., & Nambiar, S. (2009). Quality in Global Software Development Projects: A Closer Look at the Role of Distribution. *2009 Fourth IEEE International Conference on Global Software Engineering*, 163–172. https://doi.org/10.1109/ICGSE.2009.24

[14] Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, *34*(11), 131–133. https://doi.org/10.1109/2.963450

[15] Conchuir, E. O., Holmstrom, H., Agerfalk, P. J., & Fitzgerald, B. (2006). Exploring the Assumed Benefits of Global Software Development. *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*, 159–168. https://doi.org/10.1109/ICGSE.2006.261229

[16] Dullemond, K., Gameren, B. van, & Solingen, R. van. (2012). Supporting distributed software engineering in a fully distributed organization. *2012 5th International Workshop on Co-Operative and Human Aspects of Software Engineering (CHASE)*, 30–36. https://doi.org/10.1109/CHASE.2012.6223017

[17] Edwards, H. K., & Sridhar, V. (2003). Analysis of the effectiveness of global virtual teams in software engineering projects. *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of The*, 9 pp.-. https://doi.org/10.1109/HICSS.2003.1173664

[18] Ehrlich, K., Helander, M., Valetto, G., Davies, S., & Williams, C. (n.d.). *1An Analysis of Congruence Gaps and Their Effect on Distributed Software Development*. 11.

[19] Espinosa, J., Slaughter, S., Kraut, R., & Herbsleb, J. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *J. of Management Information Systems*, *24*, 135–169. https://doi.org/10.2753/MIS0742-1222240104

[20] Fairley, R. E. D., Bourque, P., & Keppler, J. (2014). The impact of SWEBOK Version 3 on software engineering education and training. *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE T)*, 192–200. https://doi.org/10.1109/CSEET.2014.6816804

[21] Feldt, R., Torkar, R., Angelis, L., & Samuelsson, M. (2008). Towards individualized software engineering: Empirical studies should collect psychometrics. *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '08*, 49–52. https://doi.org/10.1145/1370114.1370127

[22] FitzGerald, A. (2020). *The Impact of Human Factors in Distributed Software Engineering* [Unpublished manuscript.]. University of Wisconsin - Platteville.

[23] Ford, D., Storey, M.-A., Zimmermann, T., Bird, C., Jaffe, S., Maddila, C., Butler, J. L., Houck, B., & Nagappan, N. (2020). A Tale of Two Cities: Software Developers Working from Home During the COVID-19 Pandemic. *ArXiv:2008.11147 [Cs]*. http://arxiv.org/abs/2008.11147

[24] Fuks, H., Raposo, A., Gerosa, M. A., Pimental, M., & Lucena, C. (2007). *The 3C collaboration model* (pp. 637–644). https://doi.org/10.4018/978-1-59904-000-4.ch097

[25] Gumm, D. C. (2006). Distribution Dimensions in Software Development Projects: A Taxonomy. *IEEE Software*, *23*(5), 45–51. https://doi.org/10.1109/MS.2006.122

[26] Herbsleb, J. D. (2007). Global Software Engineering: The Future of Socio-technical Coordination. *Future of Software Engineering (FOSE '07)*, 188–198. https://doi.org/10.1109/FOSE.2007.11

[27] Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2000). Distance, dependencies, and delay in a global collaboration. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work - CSCW '00*, 319–328. https://doi.org/10.1145/358916.359003

[28] International Organization for Standardization. (2010). *Ergonomics of human-system interaction—Part 210: Human-centred design for interactive systems* (ISO Standard No. 9241-210:2010). https://www.iso.org/standard/52075.html

[29] International Organization for Standardization. (2015). *Software Engineering—Guide to the software engineering body of knowledge (SWEBOK)* (ISO/IEC Standard No. 19759:2015). https://www.iso.org/standard/67604.html

[30] Jabangwe, R., Börstler, J., & Petersen, K. (2015). Handover of managerial responsibilities in global software development: A case study of source code evolution and quality. *Software Quality Journal*, *23*(4), 539–566. https://doi.org/10.1007/s11219-014-9247-1

[31] Jalali, S., & Wohlin, C. (2010). Agile Practices in Global Software Engineering—A Systematic Map. *2010 5th IEEE International Conference on Global Software Engineering*, 45–54. https://doi.org/10.1109/ICGSE.2010.14

[32] Jiménez, M., Piattini, M., & Vizcaíno, A. (2009). Challenges and Improvements in Distributed Software Development: A Systematic Review. *Advances in Software Engineering*, *2009*, 1–14. https://doi.org/10.1155/2009/710971

[33] Jolak, R., Wortmann, A., Chaudron, M., & Rumpe, B. (2018). Does Distance Still Matter? Revisiting Collaborative Distributed Software Design. *IEEE Software*, *35*(6), 40–47. https://doi.org/10.1109/MS.2018.290100920

[34] Lenberg, P., Feldt, R., & Wallgren, L.-G. (2014). Towards a behavioral software engineering. *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, 48–55. https://doi.org/10.1145/2593702.2593711

[35] Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., Kuhrmann, M., & Ebdrup, A. (2018). Virtual by Design: How a Work Environment can Support Agile Distributed Software Development. *2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE)*, 97–106.

[36] Mishra, D., & Mishra, A. (2009). Effective communication, collaboration, and coordination in eXtreme Programming: Human-centric perspective in a small organization. *Human Factors and Ergonomics in Manufacturing*, *19*(5), 438–456. https://doi.org/10.1002/hfm.20164

[37] Misra, S., Colomo-Palacios, R., Pusatlı, T., & Soto-Acosta, P. (2013). A discussion on the role of people in global software development. *Tehnicki Vjesnik*, *20*, 525–531.

[38] Misra, S., & Fernández-Sanz, L. (2011). Quality Issues in Global Software Development. *ICSEA 2011: The Sixth International Conference on Software Engineering Advances*, 325–330.

[39] Morita, P. P., & Burns, C. M. (2014). Understanding 'interpersonal trust' from a human factors perspective: Insights from situation awareness and the lens model. *Theoretical Issues in Ergonomics Science*, *15*(1), 88–110. https://doi.org/10.1080/1463922X.2012.691184

[40] Nguyen, T. H. D., Adams, B., & Hassan, A. E. (2016). Does Geographical Distance Effect Distributed Development Teams: How Aggregation Bias in Software Artifacts Causes Contradictory Findings. *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 412–423. https://doi.org/10.1109/ISSRE.2016.36

[41] Nguyen, T., Wolf, T., & Damian, D. (2008). Global Software Development and Delay: Does Distance Still Matter? *2008 IEEE International Conference on Global Software Engineering*, 45–54. https://doi.org/10.1109/ICGSE.2008.39

[42] Olariu, C., & Aldea, C. C. (2014). Managing Processes for Virtual Teams – A BPM Approach. *Procedia - Social and Behavioral Sciences*, *109*, 380–384. https://doi.org/10.1016/j.sbspro.2013.12.476

[43] Olson, G. M., & Olson, J. S. (2000). Distance Matters. *Human–Computer Interaction*, *15*(2–3), 139–178. https://doi.org/10.1207/S15327051HCI1523_4

[44] Omoronyia, I., Ferguson, J., Roper, M., & Wood, M. (2010). A review of awareness in distributed collaborative software engineering. *Software: Practice and Experience*, *40*(12), 1107–1133. https://doi.org/10.1002/spe.1005

[45] Pierce, R., & St.Amant, K. (2011). Working from home in a globally distributed environment. *SIGDOC'11 - Proceedings of the 29th ACM International Conference on Design of Communication*. https://doi.org/10.1145/2038476.2038519

[46] Pirzadeh, L. (2010). *Human Factors in Software Development: A Systematic Literature Review* [Master's thesis, Chalmers University of Technology]. https://hdl.handle.net/20.500.12380/126748

[47] Prikladnicki, R., Audy, J. L. N., & Evaristo, R. (2003). Global software development in practice lessons learned. *Software Process: Improvement and Practice*, *8*(4), 267–281. https://doi.org/10.1002/spip.188

[48] Rajpal, M. (2018). Effective Distributed Pair Programming. *2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE)*, 6–10. https://dl.acm.org/doi/10.1145/3196369.3196388

[49] Ralph, P., Baltes, S., Adisaputri, G., Torkar, R., Kovalenko, V., Kalinowski, M., Novielli, N., Yoo, S., Devroey, X., Tan, X., Zhou, M., Turhan, B., Hoda, R., Hata, H., Robles, G., Fard, A. M., & Alkadhi, R. (2020). Pandemic Programming: How COVID-19 affects software developers and how their organizations can help. *Empirical Software Engineering*, *25*(6), 4927–4961. https://doi.org/10.1007/s10664-020-09875-y

[50] Reed, A. H., & Knight, L. V. (2010). Effect of a virtual project team environment on communication-related project risk. *International Journal of Project Management*, *28*(5), 422–427. https://doi.org/10.1016/j.ijproman.2009.08.002

[51] Richardson, I., Casey, V., McCaffery, F., Burton, J., & Beecham, S. (2012). A Process Framework for Global Software Engineering Teams. *Information and Software Technology*, *54*(11), 1175–1191. https://doi.org/10.1016/j.infsof.2012.05.002

[52] Šmite, D., Moe, N. B., & Torkar, R. (2008). Pitfalls in Remote Team Coordination: Lessons Learned from a Case Study. In A. Jedlitschka & O. Salo (Eds.), *Product-Focused Software Process Improvement* (pp. 345–359). Springer. https://doi.org/10.1007/978-3-540-69566-0_28

[53] Šmite, D., Wohlin, C., Galviņa, Z., & Prikladnicki, R. (2014). An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, *19*(1), 105–153. https://doi.org/10.1007/s10664-012-9217-9

[54] Stray, V., & Moe, N. B. (2020). Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. *Journal of Systems and Software*, *170*, 110717. https://doi.org/10.1016/j.jss.2020.110717

[55] 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. https://doi.org/10.1016/j.jss.2018.06.028

## Acknowledgements

This paper is adapted from a literature review written for Dr. Dianne McMullin's Human Factors Engineering course in the fall of 2020 [22]. This final version was submitted for Dr. Joshua Yue's Senior Seminar class in the spring of 2021.