Complexity Estimation for Distributed Software Development Using SRS

Apurva Agarwal
Department of Computer Engineering & Applications
GLA UNIVERSITY MATHURA
apurvag4@gmail.com

Abstract: In recent years distributed software development has gathered pace. Many development programmes now require asynchronous coordination several time zones apart between geographically dispersed teams. Owing to considerations such as effort spent on team development and technology sharing, software complexity calculation for such projects is difficult to establish a software system infrastructure that can be efficiently spread and minimizes cross-site connectivity, enabling communication between remote teams working on interrelated sections of the architecture and their day-to-day interactions. A globally distributed software project faces, in addition to the technological complexity, an additional complexity contributed by the different supporting parameters. This paper defines these parameters and attempts to estimate all the complexities described above and incorporate these two measures to obtain the corresponding overall measure of complexity during the requirement analysis phase.

1. Introduction
Geographic distribution of software development projects has been a requirement in recent years, and the software companies are now responsible for shifting their development teams to localized areas where qualified workers are more easily accessible, hence taking benefits of political and economic factors. The situation where a software company's numerous teams are spread outside a nation's limits is called Distributed Software Development. To produce a better quality of software products at low costs, as compared to co-located software projects, the main aim of this type of production criterion is to maximize capital.

Sahay (UiO) defines DSD as “Software work undertaken at geographically distributed locations across national boundaries in a coordinated fashion involving synchronous (real time) and asynchronous interaction”.

During the software lifecycle, Distributed Software Production enables team members to be located in multiple remote locations, creating a network of sub-teams that are geographically distributed in nature. Such teams may belong to the same organisation or there may be collaboration/outsourcing between various organisations. Face-to-face interaction between team members is a requirement when working on a distributed project and it involves the use of various technologies to promote communication and coordination [9].

For companies today, the distributed production of software systems and their implementations is increasingly growing in importance. There are various explanations why the trend of operating in a distributed environment should be embraced by an entity. It includes availability of a greater labour
pool and a good combination of skills and experiences, advantages in cost, and longer working hours [4].

A good development team for software has rich formal and informal interactions between its team members. It has a familiar organisational culture, required coordination and effective control; providing a good combination of technical skills and appropriate experience and known project-relevant tools and technologies. DSD introduces new challenges and requirements to the product development process as it potentially threats all of the above mentioned ideal properties.

The aim of this paper is to define and then formulate different parameters that affect the software product cost in a globally distributed setting.

2. Related Work
Jorge Cardoso [1] discussed the requirements to develop a measure to analyze the complexity of processes in distributed software development. He stated that high complexity in a process is a key to bad understandability resulting in more errors, defects, and exceptions and hence lead processes to require more time to develop, test, and maintain. Therefore, avoiding excessive complexity becomes an essential part in software development.

Threats to communication, control and coordination in distributed development were considered by the authors in [2] due to Temporal Distance, Geographical Distance, and Socio-Cultural Distance.

Dr. Ashish Sharma and Dr. D.S Kushwaha proposed a complexity measure [16] based on the factors derived from SRS Document. Even before the analysis and design phases of SDLC, the measure can estimate the software complexity efficiently. But the approach is limited to centralized software development.

R Britto et.al in [15] developed a process that helps in developing systematically onboarding strategies.

Based on four key explanations for estimating the cost and effort of software development, the Function Points (FP) system approach was used in [35] by RS Dewi et.al.

Considering Ricardo Britto et.al in [36], the findings of a systematic literature review for Global Software Development (GSD) effort estimation were presented so that the researchers and practitioners could have a clear understanding of the current state of the art about GSD effort estimation. Their outcomes indicated that an improvement on effort estimation in GSD can be made to the current state of the art.

M Shameem et.al researched [37] in order to explore and prioritize the challenges of scaling agile practices in the DSD setting. Lack of efficient communication, management responsibilities, sharing of information, etc., are described as the most critical challenges for scaling agile methods by organisations.

Through [40] the authors contributed to the specification of the system and the analysis that has the ability to assist in mapping communication patterns for global software development. In addition, this platform includes communication evaluation interfaces and intercommunications from automated engineering instruments as bots.

3. Objective
The main aim here is to define and then formulate various factors that affect the software product's cost in DSD. The complexity of these factors is stated as Supportive Complexity (SC), which is then added to the technical complexity determined in the requirement analysis phase using RDPDC [12]. The given measure of complexity gives an approximate value of the total complexity of a software product (using SRS) to assist the project managers and analysts and the concerned development teams, who are greatly influenced by the dynamic behavior of DSD.

4. Parameters That Affect Complexity in DSD
[2], [8], [19], [21], [22] discuss the various parameters that affect the complexity of a globally distributed software project. These parameters are shown below (Figure 1)
5. Proposed Measure

The proposed framework for estimating complexity of software projects in distributed environment is shown in the figure below:

![Complexity Estimation for Distributed Software Development Using SRS](image)

**Figure 2:** Overall framework for estimating CDSD

### 5.1 Technical Complexity

This complexity is estimated by adjusting the IRBC measure, as suggested in [16], in order to respond to the differences faced in DSD. The key benefit of using IRBC is to achieve the complexity of the software product using SRS in the requirement analysis phase itself.
5.1.1 Estimation of Complexity of Input / Output (IOC)

\[ IOC = Input\ _Complexity + Output\ _Complexity + Supportive\ _Complexity \]

5.1.2 Functional Requirement-Based Complexity Estimation (FR)

\[ FR = \sum_{a=1}^{n} Functionality_a \sum_{b=1}^{3} SPF_{ab} \]

5.1.3 Non-Functional Requirement-Based Complexity Estimation (NFR)

\[ NFR = \begin{cases} \sum_{a=1}^{6} (Attribute)_{a}^* \sum_{b=1}^{m} (Sub - attribute)_{ab} \\ 1 \ 	ext{otherwise for inbuilt quality attributes} \end{cases} \]

5.1.4 Estimation of Product Requirement Complexity (PRC)

\[ PRC = FR + NFR \]

5.1.5 Component (Product) Complexity Estimation (CC)

\[ CC = IOC\ *\ PRC \]

5.1.6 Personal Complexity Attributes Estimation (MPCA)

\[ PCA_i = \sum_{a=1}^{5} (Attribute)_{value}^{aib} \]

\[ MPCA = \sum_{i=1}^{s} PCA_i / s \]

Where \( s \) is the no. of sites of an organisation

5.1.7 Design Limitations Imposed Estimation (DLI)

\[ DLI = \sum_{a=0}^{5} (Limitation\ _Type)_{a} \sum_{b=0}^{6} (Number)_{ab} \]

5.1.8 Interface Complexity Estimation (IC)

\[ IC = \sum_{a=0}^{5} (External\ _Interface)_{a} \sum_{b=0}^{n} (Number)_{ab} \]

5.1.9 Product Deployment Location Complexity Estimation (PDLC)

\[ PDLC = \sum_{a=1}^{m} (LC)_{a} \sum_{b=0}^{n} (UCC)_{ab} \]

5.1.10 Product Feature Complexity Estimation (PFC)

\[ PFC = \sum_{a=1}^{n} (Feature)_{a} \]
5.1.11 Estimation of Requirement-Based Distributed Product Development (RDPDC)

\[ \text{RDPDC} = ((\text{CC} \times \text{MPCA}) + \text{DLI} + \text{IC} + \text{PFC}) \times \text{PDLC} \]

5.2 Supportive Complexity

Different additional requirement parameters given in Figure 1 add supportive complications to the overall estimate of the complexity of the software product being produced in DSD, relative to the complexity of software in centralized production.

As shown in Figure 2, the fusion of the complexities determined by Predictor Variables (PVC) and Control Variables (CV) gives the Supportive Complexity (SC). The estimation of these complexities is described in section 5.2.1 and 5.2.2.

5.2.1 Predictor Variables Complexity (PVC): The model suggested in [27] by H K Edwards & V Sridhar, uses separate predictor variables shown in Figure 2. The variables are identified after analyzing the factors that influence the efficiency and output of globally distributed virtual teams participating in the software development life cycle requirements specification process. Using heterogeneous teams scattered all over the world, the paper simulated the realities of the multinational software development environment.

Now, in order to estimate PVC, the proposed measure is using the Predictor Variables defined by the authors in [27] and the corresponding 38 measurement items used in order to prove the construct validity as they are very well covering the parameters identified in section 4 and their impact over the complexity of a DSD project.

Requirements analysis phase is a critical phase in software engineering. The effectiveness or failure of the software project is decided by the compliance of the specifications especially in case of distributed development that presents several new difficulties as well as the rise of fundamental difficulties [32] [33]. So, right from the creation of the first version of SRS by the specification team, review and adaptation by the creation team, to the creation of the last version of SRS by the specification team itself, and the allocation of tasks and their approval by the team members at remote sites requires team members to communicate effectively and work collaboratively in order to obtain a good SRS apart from several differences, be it time, language, cultural, team size or technical. Finally, “virtuality requires trust to make it work: Technology on its own is not enough” [34].

The proposed measure suggests the calculation of the prediction variables by taking the evaluation of the measurement items by the remote site team analysts during the final stages of the SDLC’s requirement elicitation and analysis phase. For the analysis to be conducted by the controlling location, the evaluations of the team analysts can be made by online GUIs accessible at the organisational sites. On a seven point Likert-type scale, all the items should be calculated as in [27] and then the mean rating of each variable for each site k is calculated as:

\[ \text{MRV}_{kp} = \frac{\sum_{a=1}^{i} \text{RMI}_{ap}}{n} \]

Where \( \text{MRP} \) refers to the mean rating of the \( p^{th} \) variable, \( \text{RMI}_{ap} \) is the \( p^{th} \) variable's \( a^{th} \) measurement item rating and \( i \) is the no. of measurement items for \( p^{th} \) variable concerned. Obviously, the max value of these measures will be 7. Thus, the total value of \( \text{MR} \) for \( s \) no. of sites can be summed up as:

\[ \text{MRV} = \frac{\sum_{k=1}^{s} \sum_{p=1}^{7} \text{MR}_{kp}}{s} \]

So, as per the range of the value of \( \text{MRV} \) a count value is assigned which gives the measure of PVC as shown in the table below:
Table 1: Estimation of PVC Count

| Mean Rating Value (MRV) Range | PVC Count |
|------------------------------|-----------|
| 1 – 7                        | 7         |
| 8 – 14                       | 6         |
| 15 – 21                      | 5         |
| 22 – 28                      | 4         |
| 29 – 35                      | 3         |
| 36 – 42                      | 2         |
| 42 - 49                      | 1         |

5.2.2 Control Variables Complexity (CCV): Control is the method to fulfill the objectives, strategies, norms, or standards of quality defined by an organisation. As seen in Fig 1, this is one of the parameters determined in section 4. In essence, the proposed measure deals with the two major forces influencing an organisation's control; they are Project Management and Instability/ Uncertainty. Using the project management variable PMGMT and an Uncertainty variable UC, both variables are calculated separately.

In order to accomplish particular objectives and goals, project management is the practice of preparing, coordinating, obtaining, and handling resources, and for this, an organisation needs well-defined procedures to help fulfill client and quality expectations.

As Dr. Edwards Deming, Quality Management theorist, consultant and author once said, "If you can't describe what you are doing as a process, you don't know what you are doing".

The SEI's Capability Maturity Model has five stages of organisational 'maturity' in terms of processes of software development that assess efficiency in the delivery of quality software. The proposed measure, as per the SEI-CMM Level, assigns a count value for the organisation and considers this value as the measure of PMGMT as shown in the figure below:

Table 2: Estimation of PMGMT Count

| SEI-CMM Level | PMGMT Count |
|---------------|-------------|
| Optimizing    | 1           |
| Managed       | 2           |
| Defined       | 3           |
| Repeatable    | 4           |
| Initial       | 5           |

Now, [26] claims that in the essence of globally distributed software projects, lack of knowledge of certain basic factors inherited makes developers fail and innovate new ways to survive. The paper summarizes various risks and their various negative effects and formulates the specifics of certain projects and their effects on the project success for managing uncertainty. Taking into account the length of the list of global threats and the difficulty of risk analysis, a method is built to computerize project outcome forecasts, appropriately labeled as a "risk barometer", for project managers to assess global project threats, taking into account the likelihood of occurrence and potential negative effects that can be correlated with past data of other projects. The instrument forecasts the amount of risk exposure for a hazard of scale \([0; 1500]\).

Here the measure proposed divides the scale into 5 ranges and gives each such range a count value and considers this value as the measure of UC, as shown in the Table 3.

Thus it is appropriate to sum up the total complexity by control variables as:

\[ CCV = PMGMT + UC \]

Hence, the overall Supportive Complexity can be measured as:

\[ SC = PVC + CCV \]

And thus, for a project developed in a distributed setting, the final complexity can be given as:
CDSD = RBDDC + SC

| Level of Risk Exposure | UC Count |
|------------------------|----------|
| 0 – 300                | 1        |
| 301 – 600              | 2        |
| 601 – 900              | 3        |
| 901 – 1200             | 4        |
| 1201 – 1500            | 5        |

### Table 3: Estimation of UC Count

6. Robustness Analysis of the Proposed Measure:
In order to analyze the robustness of CDSD, it has been evaluated against the 9 Weyuker's [5] properties. The results are shown in the Table 4:

| Property no. | Lines of Code | Cyclomatic Complexity | Halstead Measure | Dataflow Complexity | Cognitive Complexity | IRBC | Cognitive Information Complexity | CDSD |
|--------------|---------------|-----------------------|------------------|---------------------|----------------------|------|---------------------------------|------|
| 1            | Yes           | Yes                   | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 2            | Yes           | No                    | Yes              | No                  | Yes                  | Yes  | Yes                             | Yes  |
| 3            | Yes           | Yes                   | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 4            | Yes           | Yes                   | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 5            | Yes           | Yes                   | No               | No                  | Yes                  | Yes  | Yes                             | Yes  |
| 6            | No            | No                    | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 7            | No            | No                    | No               | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 8            | Yes           | Yes                   | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |
| 9            | No            | No                    | Yes              | Yes                 | Yes                  | Yes  | Yes                             | Yes  |

The application of the proposed CDSD measure has been explained in the next section.

7. Application of CDSD on Bankers Algorithm
Name of Program/ Module: Bankers Algorithm
Inputs: No. of processes and resources, claim and allocation table and maximum units of each resource.
Output: Safe process execution.
Input Complexity (IC) = 5*1*1 = 5; (Source: Console Device; Type: Text)
Output Complexity (OC) =1*1*1 = 1; (Source: Console Device; Type: Text)
Storage Complexity (SC) = 1; (Local storage)
IOC = IC + OC + SC = 5 + 1 + 1 = 7; Funcationality: Safe resource allocation to the processes in execution;
Sub Processes: Initialization, process selection for safe execution, process execution & release resources;
Functional Requirements (FR) = 1*4 = 4;
For Non-Functional Requirements (NFR)
- No. of Attributes: Functionality and Usability
- No. of Sub-attributes: Accuracy and Operability
NFR = 2;
Product Requirement Complexity (PRC) = FR + NFR = 4 + 2 = 6;
Component Complexity (CC) = IOC * PRC = 7 * 6 = 42;
No. of development sites (s) = 2;
MPCA₁ = 1.17; (Programmer’s capability is Low)
MPCA₂ = 0.90; (Programmer’s capability is High)
MPCA = (1.17 + 0.90)/ 2 = 1.035;
Design Limitations Imposed: DLI = 0;  
Interface Complexity: IC = 0;  
User Class Complexity: UCC = 1: Casual End User;  
Product Feature Complexity: PFC = 0;  
PDLC = 1*1 = 1;  
RDPDC = ((42*1.035) +0+0+0)*1 = 43.47 cu;  
For Site 1:

| Predictor Variables | Measurement Items (MI) | Mean Rating Value (MRV) | Max. Mean Rating Value |
|----------------------|------------------------|-------------------------|------------------------|
| EASTEC               | 2                      | 6                       | 7                      |
| STRPRO               | 1                      | 4                       | 7                      |
| TMEDIF               | 4                      | 5                       | 7                      |
| TRUST                | 4                      | 6                       | 7                      |
| ACADTM               | 3                      | 5                       | 7                      |
| CULTM                | 3                      | 4                       | 7                      |
| SIZETM               | 1                      | 6                       | 7                      |

Total Mean Rating (TMR₁) = 36;  
For Site 2:

| Predictor Variables | Measurement Items (MI) | Mean Rating Value (MRV) | Max. Mean Rating Value |
|----------------------|------------------------|-------------------------|------------------------|
| EASTEC               | 2                      | 6                       | 7                      |
| STRPRO               | 1                      | 5                       | 7                      |
| TMEDIF               | 4                      | 5                       | 7                      |
| TRUST                | 4                      | 6                       | 7                      |
| ACADTM               | 3                      | 5                       | 7                      |
| CULTM                | 3                      | 6                       | 7                      |
| SIZETM               | 1                      | 5                       | 7                      |

Total Mean Rating (TMR₂) = 38;  
Thus, MRV = (36+ 38)/ 2 = 37;  
PVC = 2;  
Let the SEI-CMM Level of the organization be Managed, PMGMT = 2;  
Considering the threat of Lacking Experience and Expertise in outsourcing Projects, let UC = 3;  
CCV = PMGMT + UC = 5;  
SC = PVC + CCV = 7;  
CDSD = RDPDC + SC = 43.47 + 7 = 50.47 cu;  

8. Results  
The proposed measure CDSD has been applied over ten well known algorithms and their results are compared and analyzed against the several existing ones like LOC, McCabe [14], IRBC [16], Halstead [6], CFS [41] and KLCID. Also, the complexity for ‘Email Classifier’ project is calculated using CDSD measure.  
CDSD depends on the problem criteria and the additional parameters that were held constant for all 10 problems, such as connectivity, collaboration and control between team members. Thus, the IRBC plot, which also relies on the problem specifications, is very much compatible to CDSD (Figure 3).
Table 7: Estimated Complexity Values for Existing Measures

| Problem Name       | LoC | Cyclometric Complexity | CFS | IRBC | KLCID | Halstead Measure | Proposed Measure (CDSD) |
|--------------------|-----|------------------------|-----|------|-------|------------------|-------------------------|
| Banker’s Algorithm | 123 | 22                     | 125 | 49.14| 3.08  | 63.875           | 52.47                   |
| DES Algorithm      | 503 | 74                     | 581 | 98.28| 4.16  | 167.27           | 95.94                   |
| FCFS               | 36  | 3                      | 10  | 29.25| 3.00  | 13.60            | 34.875                  |
| LCS                | 43  | 12                     | 78  | 35.10| 3.15  | 53.125           | 40.05                   |
| MST                | 40  | 13                     | 70  | 32.76| 2.47  | 47.28            | 37.98                   |
| RSA                | 81  | 17                     | 73  | 37.44| 3.24  | 56.25            | 42.12                   |
| Scheduling Algorithms | 211 | 41                     | 365 | 80.09| 3.97  | 142.50           | 88.695                  |
| Sorting Algorithms | 54  | 11                     | 45  | 21.06| 2.35  | 74.00            | 27.63                   |
| BFS and DFS        | 146 | 36                     | 263 | 58.50| 3.05  | 84.525           | 60.75                   |
| Sudoku             | 114 | 33                     | 596 | 28.08| 2.54  | 47.294           | 33.84                   |

Further, the measure was applied on an industrial project "Email.Classifier", where the final CDSD was estimated to be \textbf{6478.45 cu.}
Figure 5: Overall comparison between CDSD and other measures

9. Conclusion
This paper has investigated some factors that are affecting the performance and complexity in DSD. Through this paper, an attempt has been made to suggest a qualitative approach for estimating the difficulty of a project being developed in a distributive environment.

10. Motivation
There are several software complexity measures for estimating the difficulty while working in a centralized development environment. But, the same is not true in case of Distributed Software Development. Any framework or methodology for estimating the complexity in a distributed environment has not yet been proposed. So working on such significant area seems to be both challenging and interesting.

11. References
[1] Cardoso J 2005 About The Complexity of Teamwork and Collaboration Processes IEEE/IPSJ (SAINT 2005 Workshops) pp 218-221
[2] Agerfalk P J, Fitzgerald B, Holmstrom H, Lings B, Lundell B and Conchuir O E 2005 A Framework for Considering Opportunities and Threats in Distributed Software Development Proc. of Int. Workshop on Distributed Software Development (DiSD) pp 47-61
[3] Herbsleb J D and Grinter R E 1999 Splitting the Organization and Integrating the Code: Conway’s Law Revisited Proc. of the 21st Int. Conf. on Software Engineering (ICSE’99) (ACM Press, New York) pp 85-95
[4] Ramasubbu N and Balan R K 2007 Globally Distributed Software Development Project Performance: An Empirical Analysis Proc. of 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE ’07) pp 125–134
[5] Weyuker E 1988 Evaluating Software Complexity Measure IEEE Transaction on Software Engineering vol 14 no. 9 pp 1357-1365
[6] Halstead M H 1977 Elements of Software Science Elsevier North, New York
[7] Slaughter S A, Harter D E, and Krishnan M S 1998 Evaluating the Cost of Software Quality Communications of the ACM vol 41 pp 67-73
[8] Sengupta B, Chandra S and Sinha V 2006 A Research Agenda for Distributed Software Development ICSE’06 (Shanghai, China) pp 731-740
[9] Prikladnicki R, Audy J, Evaristo R 2003 Distributed Software Development: Toward an Understanding of the Relationship between Project Team, Users and Customers Proc. of ICEIS pp 417-423
[10] IEEE Computer Society 1994 *IEEE Recommended Practice for Software Requirement Specifications* New York

[11] Halstead M H 1977 *Elements of Software Science* Elsevier North, New York

[12] Sharma A and Kushwaha D S 2012 Applying Requirement Based Complexity for the Estimation of Software Development and Testing Effort Natarajan Meghanathan, et al. (Eds.): ITCS, SIP, JSE-2012, CS & IT 04 pp 393–415

[13] Klemola T and Rilling J 2003 A Cognitive Complexity Metric based on Category Learning *Proc. Of the 2nd IEEE Int. Conf. on Cognitive Informatics (ICCI-03)* pp 103-108

[14] McCabe T H 1976 A Complexity Measure *IEEE Transaction on Software Engineering* vol 2 no. 6 pp 308 – 320

[15] Britto R, Smite D, Damm LO, Börstler J 2020 Evaluating and Strategizing the Onboarding of Software Developers in Large-Scale Globally Distributed Projects *J. of Systems and Software* vol 169

[16] Sharma A, Kushwaha DS 2010 A Complexity Measure Based on Requirement Engineering *Document JCSE UK*

[17] Curtis B 2000 The Global Pursuit of Process Maturity *IEEE Software* vol 17 pp 76-78

[18] Herbsleb J and Moitra D 2001 Global Software Development *IEEE Software* vol 18 pp 16-20

[19] Slaughter SA, Harter DE and Krishnan MS 1998 Evaluating the Cost of Software Quality *Communications of the ACM* vol 41 pp 67-73

[20] Nasscom-McKinsey 2002 NASSCOM-McKinsey Report *National Association of Software and Service Companies* (New Delhi)

[21] Herbsleb JD 2007 Global Software Engineering: The Future of Socio-technical Coordination *Proc. of Future of Software Engineering (FOSE)* pp 188-198

[22] Carmel E and Agarwal R 2001 Tactical Approaches for Alleviating Distance in Global Software Development *IEEE Software* pp 22-29

[23] Nordio M, Estler HC, Meyer B, Tschannen J, Ghezzi C and Nitto E Di 2011 How do Distribution and Time Zones affect Software Development? A Case Study on Communication *IEEE 6th Int. Conf. on Global Software Engineering* pp 176-184

[24] Damian D and Moitra D 2006 Global Software Development: How Far Have We Come? *IEEE Software* vol 23 no.5 pp 17-19

[25] Smite D 2007 Project Outcome Predictions: Risk Barometer Based on Historical Data *In Proc. of the Int. Conf. on Global Software Engineering (ICGSE) by IEEE Computer Society, Germany* pp.103-112

[26] Smite D, Borzovs J 2008 Managing Uncertainty in Globally Distributed Software Development Projects *University of Latvia, Computer Science and Information Technologies* vol 733 pp 9-23

[27] Edwards HK and Sridhar V 2003 Analysis of the Effectiveness of Global Virtual Teams in Software Engineering Projects *36th Annual Hawaii Int. Conf. on System Sciences Proceedings of the, Big Island, HI, USA* (doi: 10.1109/HICSS.2003.1173664) p 9

[28] Ebert C and Neve P De 2001 Surviving Global Software Development *IEEE Software* pp 62-69

[29] Carmel E and Agarwal R 2011 Tactical Approaches for Alleviating Distance in Global Software Development *IEEE Software* pp 22-29

[30] Favela J and Mora F 2001 An Experience in Collaborative Software Engineering Education *IEEE Software* pp 47-53

[31] Sommerville I and Kotonya G 1998 Requirements Engineering: Processes and Techniques (Toronto: John Wiley & Sons)

[32] Damian D 2002 The Study of Requirements Engineering in Global Software Development: As Challenging As Important *Proc. of Int. Workshop on Global Software Development – ICSE* (Florida, USA)
[33] Zowghi D 2002 Does Global Software Development Need a Different Requirements Engineering Process? Proc. of Int. Workshop on Global Software Development – ICSE (Orlando, Florida, USA) pp 53-55

[34] Handy C 1995 Trust and the Virtual Organisation Haward Business Review vol 73 no. 3 pp 40-50

[35] Dewi RS and Subriadi AP 2017 A Modification Complexity Factor in Function Points Method for Software Cost Estimation Towards Public Service Application Procedia Computer Science, Elsevier

[36] Britto R, Freitas V, Usman M 2014 Effort Estimation in Global Software Development: A Systematic Literature Review in Proc. of 9th IEEE Int. Conf. on Global Software Engineering (ICGSE'14) - Shanghai, China

[37] Shameem M, Kumar RR and Kumar C 2018 Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process J. of Software - Wiley Online Library

[38] Akbar MA, Sang J, Khan AA and Mahmood S 2019 Success factors influencing requirements change management process in global software development J. of Computer ..., – Elsevier

[39] Cozzolino S 2020 The (anti) adaptive neighbourhoods. Embracing complexity and distribution of design control in the ordinary built environment J. of Environment and Planning B: Urban Analytics vol 47 Issue 2 pp 203-219

[40] L’Erario A, Gonçalves J, Fabri JA et al. 2020 CFSD: a Communication Framework for Distributed Software Development J. of the Brazilian Computer Society vol 26 p 7

[41] Wang Y and Shao J 2003 Measurement of the Cognitive Functional Complexity of Software Proc. 2nd IEEE Int’l Conf. on Cognitive Informatics (ICCI’03) (IEEE CS Press) pp 67-74