Chapter N

New Forms of Work in the Light of Globalization in Software Development

Darja Smite

Blekinge Institute of Technology, Sweden

University of Latvia and Riga Information Technology Institute, Latvia

Juris Borzovs

University of Latvia and Riga Information Technology Institute, Latvia

KEYWORDS

Global software engineering, Outsourcing, Offshoring, Forms of work, Collaboration models

ABSTRACT

Globalization in software development introduced significant changes in the way organizations operate today. Software is nowadays produced by team members from geographically, temporally and culturally remote sites. Organizations seek for benefits that global market offers and face new challenges. Naturally resistant to change, these organizations often do not realize necessity for tailoring existing methods for distributed collaboration. Our empirical investigation shows a great variety in the ways organizations distribute responsibilities across remote sites and conclude that these can be divided into two main categories: joint collaboration that requires investments in team building and independent collaboration that requires investments in knowledge management and transfer. Finally we discuss practices that are applied in industry to overcome these challenges and emphasize necessity to fully understand the pros and cons of different ways to organize distributed software projects before starting a project in this new environment.

INTRODUCTION

Recognized as the phenomenon of the 21st century (Friedman, 2005), globalization of the world economies brought significant changes to nearly all industries, including information technology (IT) and, in particular, software development. **Global software work** originates from IT outsourcing that is recognized as a natural evolution of how the global market operates today

(Minevich et al, 2005).

Tight budgets, shortage in resources and time has motivated many companies to start looking for partners outside. Accordingly, **outsourcing** and especially **offshoring** (relocation of business processes to a lower cost country) have become components of a new global paradigm that is based on the selection of appropriate and strategic technologies, skills and resources with the strongest potential and the lowest cost within the global marketplace.

The decision to source software development to an overseas firm is looked at frequently in simple economic terms - it's cheaper, and skilled labor is easier to find (Carmel et al, 2005). The list of assumed benefits of **global software engineering** (GSE) also includes the necessity of reaching mobility in resources, obtaining extra knowledge through deploying the most talented people around the world, speeding time-to-market, increasing operational efficiency, improving quality, expanding through acquisitions, reaching proximity to market and many more. However, a recent empirical investigation shows that these benefits are neither clear-cut nor can their realization be taken for-granted as the GSE literature may lead one to believe (Conchúir et al, 2006). After a decade of experimentations companies come to realize that blind cost reduction strategies tend to fail. And the reason for this is distinction between manufacturing of goods and intellectual work.

It is not a secret that manufacturing has spread globally, and even branded products are nowadays developed by emerging nations. Software industry follows this trend and leads towards mass-production of software components following standardized product-line approaches. India and China became well known phenomenally growing software development centers. Smaller nations are also competing with each other for their best deals from the world leading contractors.

However, in contradiction to manufacturing, distribution of intellectual work is not as easy as it may seem. No matter how much companies try to industrialize software development processes by developing small pieces that would be integrated into a product at the end, the process of software development significantly depend on human interaction. In contrast to other engineering disciplines, where actual development is based on stable plans and technical designs, software engineering is suffering from rapid changes in requirements throughout the development life cycle. And if a co-located team can cope with uncertainty and changes more effectively, distribution of software life cycle activities among team members separated by contextual, organizational, cultural, temporal, and geographical boundaries introduces significant difficulties in managing interdependencies. Virtual and often asynchronous environment that characterize globally distributed projects affect the way team members interact and communicate and form an iceberg of problems that are often hidden to an unfamiliar eye.

BACKGROUND

The main **challenges of global software engineering** are caused by the uniqueness of working environment and are not related to technical challenges that project managers are used to overcome. These challenges are brought by geographic, temporal and cultural distance between the global software team members. Related studies recognize that the key areas of concern and major sources of overhead caused by these barriers are concentrated around communication,

coordination and control activities (Ågerfalk et al, 2005).

Communication in globally distributed projects is troubled by temporal and geographic distance. A lack of working hour overlap leads to asynchronous interaction and sequential delays in information turnaround. Geographic distance affects the ways of interaction. Computer-mediated communication is much poorer than personal contact or even a phone conversation. However the later are costly and thus often not considered in global projects. Distance between the remote sites affects the amount of interaction too. Frequency of communication decreases with the distance among the team members (Carmel, 1999). Issues that are easily discussed and resolved at a cup of coffee in the corporate kitchen often hang in inbox for several days. Poor socialization, lack of frequent feedback, and unpredictability in communication often leads to lacking **trust** among the team members (Moe et al, 2008). Empirical study of speed and communication in globally distributed software development suggests that distributed work items appear to take about 2.5 times as long to complete as similar items where all the work is colocated due to interdependencies between the people involved in the project (Herbsleb et al, 2003). Moreover, empirical studies indicate an exponential decrease in productivity as dispersion increases. Thus, global environment is a challenge for developing an effectively performing software team.

Coordination of work between the remote sites is also challenged by separation. Task deconstruction and allocation becomes more complex due to lacking proximity and increasing complexity of maintaining interdependencies between the remote sites working together. Research agenda demand exploring whether product decoupling can successfully reduce coordination challenges at the same time without sacrificing the essence of teamwork (Powell et al, 2004). On one hand, related research suggests that coordination and performance will improve, if distributed team members are decoupled, which will automatically decrease the necessity to synchronize efforts and minimize the amount of communication needed to perform the tasks (Ramesh et al, 2002). This can be achieved through e.g. task decomposition using feature-oriented task allocation strategies and thus introducing coordination through standardization of low-level work processes as suggested by McChesney and Gallagher (McChesney et al, 2004). On the other hand, Conchúir et al (2006) found that, while the modularization of work can be effective in reducing the required level of cross-site communications, it might also be an obstacle to the creation of a sense of cross-site teamness. Moreover, their empirical investigation shows that there are different approaches to task modularization and these may lead to advantages as well as disadvantages (Conchúir et al 2006). Another emerging trend to address coordination and communication challenges is introduction of agile methods in distributed software teams to encouraging coordination based on mutual adjustment. Empirical studies show that agile practices are found to be useful for reducing communication, coordination, and control problems that have been associated with distributed work (Holmström et al, 2006).

Control of the remote sites in globally distributed software development projects can be characterized by lack of transparency and computer-mediation of the processes. Computer-mediated, geographic and temporal separation leads to breakdown in control and can easily paralyze managers in their daily supervision. Thus global project managers often feel insecure

and tend to overly supervise the remote team members. However, lack of **trust** and direct supervision as a result of this insecurity is found to have a negative effect on team performance (Moe et al, 2008).

These are only few examples of challenges that are brought by various aspects of distance and diversity – challenges that are inherited in global software engineering environment. Though co-located or in-house software development life cycle management has already achieved certain maturity, globalization requires new processes, methods, and tools to be implemented. Solutions that have been applied in co-located teams to overcome similar problems, do not have expected outcome when team members work in separation. Thus, achieving global project success is not an easy task. Furthermore, a high increase in required resources is often caused by untimely or inadequate reaction to problems such as too overhead of coordination and communication, rigorous documentation and standardization of the processes, as well as extra travel and training of the team members that tend to grow along with an increasing attrition. Moreover, projects that face a productivity decrease caused by incapable developers from overseas are actually fewer than projects run by unprepared managers. Problems associated with distribution may jeopardize a project and thus have to be monitored on a regular basis. Managers have to be aware that different ways of organizing distributed software work can lead to more effective or less effective performance. Thus a deep understanding of alternative ways of collaboration is very important.

FORMS OF WORK IN DISTRIBUTED PROJECTS: CASE STUDY

Empirical Background

In this chapter we share experience and reflections from empirical investigation of different forms of work practiced by an international software house that provides software development outsourcing services. The investigated entity of the software house is located in Latvia and was established in the late 80s and changed its owners and structure several times. It has been oriented towards the international market and achieved ISO 9001:2000 certified quality management. At the time of our empirical investigation the organization had successfully accomplished more than 200 projects both in Latvia, Western Europe and Scandinavia, and employed over 380 employees. For confidentiality reasons we do not disclose the name of the company.

Our conclusions discussed in this chapter are based on a survey of 38 distributed projects, field observations and interviews conducted with project managers and team members involved in distributed projects. In this chapter we illustrate different work-partitioning approaches used by the studied projects and discuss factors that affect alternatives and the effect of each form of work. At the end we share recommendations gained from practitioners.

Collaboration Models

We analyze different form of work based on investigation of remote site involvement in software development activities. In particular, our aim was to understand whether the remote sites were engaged in independent activities or worked jointly on completion of software development

activities. We differentiate two major sites involved in collaboration – the investigated organization, which we call supplier, and its customers that engage outsourcing service suppliers in distributed collaboration.

By the means of a survey we received data from 38 distributed projects indicating different **collaboration models**. We have identified 19 different models out of 38 investigated projects. In addition, we have also gathered qualitative measurements of success and failure of each model; however, our conclusions cannot be generalized due to statistical insignificance. Nonetheless, investigation of different collaboration models supports conclusions on different **forms of work** and their variations based on the level of remote site involvement (see Table 1).

Table 1. Different Forms of Work.

Nr.	Involvement of remote sites in software development activities				No of Projects
	System Analysis	Design	Implementation	Testing	
M1	jointly	jointly	jointly	jointly	3
M2	customer	customer	jointly	jointly	1
M3	customer	jointly	jointly	customer	2
M4	customer	customer	jointly	customer	1
M5	[no information]	[no information]	supplier	[no information]	2
M6	[no information]	[no information]	supplier	customer	1
M7	customer	customer	supplier	customer	5
M8	customer	customer	customer	supplier	1
M9	customer	customer	supplier	supplier	3
M10	customer	supplier	supplier	supplier	1
M11	customer	supplier	supplier	customer	5
M12	customer	customer	supplier	jointly	1
M13	customer	supplier	supplier	jointly	2
M14	jointly	jointly	supplier	supplier	3
M15	customer	jointly	supplier	jointly	2
M16	customer	jointly	supplier	customer	1
M17	customer	jointly	supplier	supplier	1
M18	jointly	supplier	supplier	supplier	1
M19	supplier	supplier	supplier	jointly	1

The investigated collaboration models can be divided into four different forms of work:

- Involvement of supplier in joint activities (M1-M4);
- Outsourcing of selected activities with some level of joint performance (M5-M11);
- **Outsourcing** without joint performance (M12-M17);
- Independent remote development (M18-M19).

Motivated by market pressures, cost saving strategies and lack of awareness of global threats, the most frequently met collaboration type is outsourcing without joint performance. Accordingly, the most frequent collaboration model is sending coding activities to a remote supplier (followed by 5 projects – M5, M6, M7) or sending one or several other activities to a remote supplier (followed by 5 projects – M8, M9, M10, M11).

There are many projects with mixed distribution of activities – joint and independent performance. There are 6 projects with dominance of the customer independence (M2, M3, M4, M12, and M16) and 8 projects with supplier independence dominating (M13, M14, M17, M18, and M19). Full outsourcing projects, especially the most extreme representatives of this type of collaboration, don't appear among the investigated projects, because there is no process distribution. Accordingly these projects do not fit the phenomenon under study. However, variations of independent remote development appear (followed by only 2 projects).

And finally, joint performance between the partners is experienced by 3 projects (M1).

Factors Affecting Lifecycle Distribution

Empirical observations and interviews with practitioners help to derive the following list of factors affecting lifecycle distribution in global projects.

- Motives for going global. The main motives for organizations to switch to remote software service supplier often determine the selection of lifecycle distribution. Accordingly, organizations aiming to decrease costs usually send independent pieces of work to remote suppliers, whereas organizations aiming to gain extra knowledge or workforce usually engage remote sites in joint performance to fill the competence gaps.
- Level of experience with outsourcing. Our investigation shows that forms of work may change over time within one project or from project to project conducted among the collaborating partners. Organizations are also resistant to allocate responsibility for a large piece of work to a remote partner, when collaboration is new.

Teamwork Challenges vs. Knowledge Transfer Challenges

By the nature of each of the forms of work, we can conclude that projects managers have to balance between distributed teamwork challenges and distributed knowledge transfer challenges dependent on the chosen collaboration model. We have gathered experience from projects and discuss different problems associated with each of these approaches in the following table.

Table 2. Problems Associated with Different Forms of Work.

Teamwork Challenges	Knowledge Transfer Challenges
<ul style="list-style-type: none"> • Lack of teamness • Uncooperative behavior • Lack of trust • Inability of synchronous work and communication due to temporal distance • Lack of specific tool support that facilitates distributed work 	<ul style="list-style-type: none"> • Unwillingness to share knowledge and experience due to fear of job loss • Lack of trust • Insufficient detail of requirements specification and task descriptions • Lack of tools that support remote access and information exchange

Lack of proximity and next-door closeness between the developers involved in the project makes it difficult to expect the remote developers acting as a joint co-located team without additional team building activities. Moe and Smite (2008) report the results of a multi-case study conducted in the same organization as described in this book. The study points out that although remote team members ought to form a joint team they often consider distribution as team separator that hinders distributed members to form a joint team. Similarly, Casey and Richardson (2006) discuss un-cooperative behavior of remote team members who use e-mail as a weapon to publicly attack fellow team members. Treinen and Miller-Frost (2007) describe how remote teams exhausted by the asynchronous work over multiple temporal zones lay blame for overall project problems on each other. At the same time, team-building activities are often withdrawn due to limited budgets and thus building **trust** without “touch” becomes a challenge (Moe et al., 2008). In result, as previously discussed, empirical studies show that distributed teams are far much less productive than co-located teams (Herbsleb et al, 2003).

On the other hand, independent performance requires far much more effort and, subsequently, the cost of documenting requirements than practiced in co-located projects (Herbsleb et al, 2005). This shall be done to avoid inconsistency, misunderstandings, and rough transfer between the phases. Computer-mediated communication troubles clarification of inconsistent requirements over the distance (Smite, 2006). Empirical studies show that remote teams experience knowledge sharing barriers, such as unwillingness to share knowledge due to fear of job loss caused by outsourcing (Casey et al, 2006). Knowledge management is also challenged by technical infrastructure and inability to access information from remote locations (Oshri et al, 2007).

RECOMMENDATIONS TO PRACTITIONERS

Areas of Improvement

Despite the popularity of global software engineering, no research could determine the exact recipe for effective outsourcing performance (Loh et al, 1995). The results of our investigation indicate that there is a great variety of ways to organize distributed work. Process distribution and

remote partner involvement decisions are related to different factors such as previous experience and reasons for outsourcing. Thus there might be no silver bullet or “one-fits-all” solutions to address the needs of every organization that is engaged in distributed software development. Moreover, the two alternative approaches – reliance on distributed teamwork and reliance on distributed knowledge transfer – have both their pros and cons. Hence, every organization shall consider the strengths and weaknesses of their distributed teams, and invest either in team building activities or knowledge management processes.

Analysis of global software engineering improvement areas leads to a conclusion that in both cases, whether it is necessity for team building or better knowledge management, improvements shall be developed and implemented across borders. This means that distributed project members shall put joint effort into process improvement initiatives. This however becomes an issue if collaboration involves team members from different organizations. Empirical studies show that supplier initiatives may face indifference from the customer site (Moe, 2008), but improvements focused only in one site may have a limited impact on overall project performance. Thus, joint effort into problem resolution is required.

Considerations

Our empirical investigation emphasizes the necessity of deliberate planning and analysis of alternatives for organizations engaging in global software engineering. Besides considering the two options of process distribution, we also recommend to consider the following issues:

- Don't underestimate the effect of diversity. Evaluate the risks associated with knowledge transition between remote locations taking into account diversity in background, education and work habits of the team members. Our experience shows that even organizations from the seemingly close locations, such as countries around Europe, may have different approaches to software engineering activities and thus, may lead to misaligned expectations and requirements of the remote sites. Thus, getting to know each other's ways of working are essential for successful collaboration.
- Pay attention to flexibility and adaptability. Diversity and inconsistency, in other words heterogeneity of the remote sites and their backgrounds may lead to various problems regarding collaboration. Despite the fact that organizations are naturally resistant to change, those that involve subcontractors should experiment and adjust the global product delivery models by decreasing the processes and interaction layers that “steal” time and consider changes focusing on improvements that would enable effective cooperation of distributed team members.

Corrective Actions

Interviews with experienced project managers and field observations form a set of practices that proved to be effective in an industrial context of the Latvian software house under study. Supplemented by related literature, our recommendations for organizing the work across remote sites are as follows.

- Establish common philosophy and approach. To mitigate diversity between the collaborating

sites, project managers shall establish common philosophy and approach. This is related to work practices, as well as perception of certain aspects of lifecycle management, such as quality, process inputs, outputs and entry/exit criteria etc., and is essential for both types of collaboration whether it will be independent or joint performance. However, organizations shall not take this recommendation formally. Establishing common project standards is not the main challenge. The most important thing is to achieve a common understanding and commitment from the remote team members on how they will collaborate together.

- Plan and perform small increments. Pessimistic project managers often lack belief in ability to perform from a far-off location. This may negatively affect sites working in a globally distributed environment. Planning and performing small increments with frequent deliverables will provide supplier a chance to demonstrate ability to perform and assure project success. This shall contribute to establishment of cognition, mutual respect and trust.
- Effective information- and knowledge-sharing mechanisms are recognized as one of the key factors determining the ability to achieve benefits of global software engineering (Holmström et al, 2006). Experienced project managers suggest temporal or permanent relocation of team members during critical project phases in order to bridge the remote sites more effectively. Our observations show that it is recommended and practiced to encourage systems analysts to travel to remote sites for more efficient requirements clarification. This recommendation is also supported by many related studies, which also call these 'liaisons' bridgeheads (Lings et al, 2007) or knowledge scouts (Dutoit et al, 2001).
- Improve performance during the project, if it is lower than expected. Ineffective global projects and remote team members performing below the expected productivity level is more than just a headache. Yet practice shows that productivity problems can be caused not only by poor performance and lack of knowledge of the supplier. Root of many problems may come from inappropriate project lifecycle management or infrastructure. Explore the actual sources of problems and mitigate them on a continuous basis.

FUTURE RESEARCH DIRECTIONS

It has been noted that communication, coordination and control are the three major challenges in distributed software development (Ågerfalk et al, 2005). It is important to understand the impact of different approaches to software project life cycle organization. In the light of empirical findings we see the need for investigating the advantages and disadvantages of the two studied approaches for organizing the work across remote sites, namely involvement of remote sites in independent or joint performance. Future research shall deepen our understanding of investments necessary to build an effective team from one hand, and the context of applicability for independent performance on the other hand. Moreover, we support Powell et al. (2004) and encourage to explore whether product decoupling can successfully reduce coordination challenges at the same time without sacrificing the essence of teamwork.

CONCLUSIONS

Global software engineering is certainly more than a temporary trend. It is a part of today's market with a stable growth in future. It requires new skills and will drive organizations to significant changes not only in respect to business forms, but also forms of work, habits, skill portfolios and corporate culture.

In this chapter we discuss results of our empirical investigation that uncovered a great variety of different ways to organize collaboration across remote locations. These collaboration types vary between full outsourcing and full partnership categorized into 4 major groups: involvement of supplier in joint activities; outsourcing of certain activities with some level of joint performance; outsourcing without joint performance; independent remote development. The major difference between different approaches is rooted in the level of supplier involvement in either independent or joint software development activities. It has been observed that joint collaboration require investments into team building and independent collaboration stresses necessity for knowledge transfer and proper work decoupling.

Due to market pressures, cost saving strategies and lack of awareness of global threats organizations often try to organize outsourcing software projects similarly to co-located projects, which inevitably leads to inconsiderate decisions regarding process distribution and failure. Moreover, naturally resistant to change, organizations involved in distributed work often do not realize the unique challenges of distributed projects till it is already too late to improve and save the project from failure. However, facing unmet expectations and hindered benefits, managers come to realize the need for process reengineering. In the study conducted by Casey and Richardson processes effective for single site development proved inadequate for a virtual software team environment (Casey et.al, 2006). Thus we emphasize necessity to fully understand the pros and cons of different ways to organize distributed software projects and timely preparations for new working environment.

REFERENCES

- Carmel E. (1999). *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, NJ, USA. Prentice-Hall
- Carmel E. & Tjia P. (2005). *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*. Cambridge University Press
- Casey V. & Richardson I. (2006). Uncovering the reality within virtual software teams. In: P. Kruchten et.al (Ed.), *International workshop on Global software development for the practitioner* (pp. 66-72). ACM: New York, USA
- Conchúir E. Ó., Holmström H., Ågerfalk P.J. & Fitzgerald B. (2006). Exploring the Assumed Benefits of Global Software Development. In P. Fernandes et.al (Ed.): *IEEE International Conference on Global Software Engineering*, (pp.159-168). IEEE Computer Society: Los Alamitos, CA, USA

- Dutoit A.H., Johnstone J. & Bruegge B. (2001). Knowledge scouts: Reducing communication barriers in a distributed software development project. In H. Jifeng et al. (Ed.): *Asia-Pacific Software Engineering Conference (APSEC)* (pp. 427-430). IEEE Computer Society: Los Alamitos, CA, USA
- Friedman T.L. (2005). *The World is Flat: Brief History of the 21st Century*. New York, US: Farrar, Straus and Giroux
- Herbsleb, J., Paulish, D.J., & Bass, M. (2005). Global software development at Siemens: Experience from nine projects. In W. Griswold et al. (Ed.): *International Conference on Software Engineering (ICSE)* (pp. 524-533). ACM, New York, NY
- Herbsleb J. D. & Mockus A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6), 481-494
- Holmström H., Fitzgerald B., Ågerfalk P. J. & Conchúir E. Ó. (2006). Agile Practices Reduce Distance in Global Software Development. *Information Systems Management*, 23(3), 7-18
- Lings B., Lundell B. Agerfalk P. J. & Fitzgerald B. (2007). A reference model for successful Distributed Development of Software Systems. In F. Paulisch et al. (Ed.): *International Conference on Global Software Engineering* (pp.130-139). IEEE Computer Society: Los Alamitos, CA, USA
- Loh L. & Venkatraman N. (1995). An empirical study of information technology outsourcing: Benefits, risks, and performance implications. In: *International Conference on Information Systems* (pp.277-288)
- McChesney I. R. & Gallagher S. (2004). Communication and Co-ordination Practices in Software Engineering Projects. *Information and Software Technology*, 46(7), 473-489
- Minevich M. & Richter F.J. (2005). Global Outsourcing Report 2005. Going Global Ventures Inc., New York | Horasis, Geneva
- Moe N. B. & Smite D. (2008). Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study. *Software Process Improvement and Practice*, 13(3) 217-231
- Oshri I., Kotlarsky J. & Willcocks L.P. (2007). Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects. *The Journal of Strategic Information Systems*, 16(1), 25-49
- Powell A., Piccoli G. & Ives B. (2004). Virtual Teams: A Review of Current Literature and Directions for Future Research. *The DATA BASE for Advances in Information Systems*, 35(1), 6-36.
- Ramesh V. & Dennis A. (2002). The Object-oriented Team: Lessons for Virtual Team from Global Software Development. In J. F. Nunamaker Jr. et al. (Ed.): *Annual Hawaii International Conference on System Sciences*, (pp. 212-221), IEEE Computer Society: Los Alamitos, CA, USA
- Smite D. (2006). Global Software Development Projects in One of the Biggest Companies in Latvia: Is Geographical Distribution a Problem. *Software Process Improvement and Practice*,

11(1), 61-76

Treinen J.J. & Miller-Frost S.L. (2006). Following the sun: Case studies in global software development. *IBM Systems Journal*, 45(4), 773-783

Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., & Conchúir, E. Ó. (2005). A Framework for Considering Opportunities and Threats in Distributed Software Development. In: *International Workshop on Distributed Software Development (DiSD)*, (pp. 47–61), Austrian Computer Society

ADDITIONAL READING SECTION

Carmel E. (1999). *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, NJ, USA. Prentice-Hall

Carmel E. & Tjia P. (2005). *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*. Cambridge University Press

Friedman T.L. (2005). *The World is Flat: Brief History of the 21st Century*. New York, US: Farrar, Straus and Giroux

KEY TERMS & DEFINITIONS

Outsourcing: Subcontracting one or more software development processes to an external supplier

Offshoring: Relocation of one or more software development processes to a lower cost country within the same organization

Global software engineering: Transition of common co-located software engineering to more complex software life cycle activities distributed among teams separated by various boundaries, such as contextual, organizational, cultural, temporal, geographical, and political.

Globally distributed software engineering: See Global software engineering.

Forms of work: Collaboration models that outline process distribution and partner involvement in global software engineering projects

Teamwork: joint involvement of remote partners in globally distributed software engineering project activities

Knowledge transfer: Explicit and implicit knowledge sharing among remote partners necessary for transition between activities in a software engineering project