Benefits of Global Software Development: The Known and Unknown

Pär J. Ågerfalk[1,2], Brian Fitzgerald[1], Helena Holmström Olsson[1,3], and Eoin Ó Conchúir[1]

[1] Lero, The Irish Software Engineering Research Centre, University of Limerick, Ireland
[2] Uppsala University, Sweden
[3] IT University, Gothenburg, Sweden
par.agerfalk@dis.uu.se, bf@ul.ie, helena.holmstrom@lero.ie, eoin.oconchuir@lero.ie

Abstract. Organizations are increasingly moving to the global software development (GSD) model because of significant benefits that can accrue. However, GSD is fraught with difficulties arising from geographical, temporal and socio-cultural distances. The emphasis in the literature to date has typically been on how to overcome these significant challenges associated with GSD. While a number of GSD benefits have been identified in the literature, there are also a number of less obvious, what we term 'unknown,' potential benefits that can accrue from GSD. Here we synthesize and integrate an overall set of potential GSD benefits and categorize them according to the organizational, team and process level to which they are most applicable. The 'unknown' includes organization benefits, such as improved resource allocation, team benefits, such as reduced coordination cost and improved team autonomy, and process benefits, such as improved documentation and clearly defined processes.

Keywords: global software development, benefits, challenges, offshoring.

1 Introduction

Global software development (GSD) is a phenomenon of increasing importance, given the perennial pressures of the need to remain profitable and competitive in the global landscape. Companies can now leverage the emergence of large multi-skilled labor forces in lower-cost economies thanks to high-speed Internet-based communication links, through which the product (software code) can be quickly transferred between development sites. India and China, in particular, offer huge multi-skilled labor forces at greatly reduced cost compared with employment markets in the US and western Europe. Other countries are also making an impact, such as Brazil, Eastern Europe and Russia, Malaysia and Vietnam.

GSD involves the three major types of distance: geographical, temporal, and socio-cultural [1]. Single teams can be separated by these distances, essentially becoming what is often termed 'virtual teams'. In other circumstances, a single team may have all of its resources co-located, but with heavy reliance on other teams at remote

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2008, LNCS 5007, pp. 1 – 9, 2008.

locations. Vast geographical distances imply the difficulty of re-locating to another of the company's sites, and not being able to stroll over to a colleague's desk to chat about an implementation issue. Temporal distance across multiple time zones reduces the number of overlapping working hours, forcing a heavier reliance on asynchronous communication technologies. Socio-cultural distance arises from the different national and organizational backgrounds of the people involved and exacerbates communication breakdown.

Major benefits have been attributed with GSD despite the challenges arising from overcoming these distances. Apart from being a potential side-effect of mergers and acquisitions, GSD is purported as enabling major benefits such as lower development costs and access to huge multi-skilled labor forces, as already mentioned. However as researchers and practitioners have focused on overcoming GSD challenges, an exhaustive inventory of potential GSD benefits has not been compiled. While some benefits have been widely acknowledged in previous research, other potential benefits are evident but, nonetheless, overlooked to a large extent. In this paper, we label these two categories of benefits 'known' and 'unknown' – and explore what each benefit might offer companies aiming to leverage GSD.

1.1 Challenges of GSD

The geographical, temporal, and socio-cultural distances affect the three major processes of software development: communication, coordination, and control [1]. In fact, communication and control problems are recognized as being the most troublesome and pervasive in software development [2].

A major challenge for GSD teams is the lack of informal communication which has been found to be an essential process in traditionally co-located development [3, 4]. Written documentation is often inadequate when resolving misunderstandings, such as misunderstandings about requirements or changes in requirement specifications. Geographical and temporal distances make it more difficult to initiate contact with colleagues at other locations. While being indispensible for enabling face-to-face contact, the cost of travel can be prohibitive. A lack of overlapping working hours can lead to delays in feedback, rendering the development process less effective. Even a time zone difference of one hour can have a major effect when combined with different typical working hours in different countries.

Socio-cultural distance can result in a fundamental difference in opinion about the nature of the software development process [5]. It can lead to misunderstandings and non-native speakers struggling to follow technical discussions, especially over the phone. A general lack of familiarity with remotely located colleagues can result in a lack of 'teamness' and a reduced sense of trust.

Moreover, the distances involved increase the complexity involved in coordinating the software development process. Software development in itself is a complex task with substantial non-routine work. The GSD environment calls for increased understanding for this complexity and the ability to focus on coordination of resources, such as distributed teams and tasks. Clearly, achieving a satisfactory result can be a challenge.

1.2 Assumptions Made About GSD

Despite well-known challenges, GSD also presents practitioners with various benefits. As pointed out above, some of these are well known, while some are not as obvious. Interestingly, the 'known' benefits, which are generally considered to be the driving forces behind GSD, all seem to apply at the organizational level. That is, they contribute to top-level organizational goals, such as cost savings and increased efficiency. Admittedly, some of the 'unknown' benefits apply at the organizational level, but in addition we see benefits that more directly affect teams and basic software development processes and tasks. We would argue that the 'unknown' benefits should also be taken into consideration and that there is a need to highlight the full spectrum of GSD benefits.

Currently there is a tendency to 'localize' GSD by attempting to reduce the geographical, temporal and socio-cultural distances involved. This approach assumes that the benefits of GSD do not fully justify truly global software development. Contrary to this, we have found teams shifting their working hours to increase the temporal overlap with remote colleagues, thereby aiming towards a 'virtual 8-hour day' [6].

However, the decision of whether or not to globalize software development activities – or indeed the inclination to either 'localize' or fully leverage GSD – should be informed by the potential benefits it offers. We argue that this decision can be better informed if both 'known' and 'unknown' benefits are taken into consideration. In what follows, we outline both 'known' and 'unknown' benefits. In the end, we provide a categorization of all benefits, using the categories (1) organizational, (2) team and (3) process/task.

2 The 'Known' Benefits of GSD

In this section we outline well-known benefits of GSD. They all seem to apply at organizational level and have been previously acknowledged in research.

2.1 Cost Savings

Perhaps the original and most sought-after benefit of GSD has been that of reduced cost of development [7]. The basis for this benefit is that companies are globalizing their software development activities to leverage cheaper employees located in lower-cost economies. This has been made possible by the deployment of cross-continental high-speed communication links enabling the instantaneous transfer of the basic product at hand: software.

The difference in wages across regions can be significant, with a US software engineer's salary being multiple times greater than that of a person with equivalent skills in (at least parts) of Asia or South America. In 2005, the annual base pay of a software engineer located in India was $10,300 [8]. However, this seems to be rising and there has been hyper-growth in local I.T. employment markets such as in Bangalore. It is our experience that companies are now looking at alternative locations which offer more acceptable attrition rates with the continued promise of cheaper labor.

2.2 Access to Large Multi-skilled Workforces

GSD provides the unprecedented possibility to leverage large pools of skilled labor by coordinating across distance [9, 10, 11, 12]. Companies have the opportunity to expand their software development activities to include the contributions of thousands of skilled workers, wherever they may be located, to form virtual global corporations [13, 11, 14].

2.3 Reduced Time to Market

A controversial benefit of GSD has been that of the 'follow-the-sun' approach, described in detail by Carmel [4]. Time zone effectiveness is the degree to which an organization manages resources in multiple time zones, maximizing productivity by increasing the number of hours during a 24-hour day that software is being developed by its teams. When time zone effectiveness is maximized to span 24 hours of the day, this is referred to as the 'follow-the-sun' development model. This is achieved by handing off work from one team at the end of their day to another team located in another time zone. The approach can aid organizations which are under severe pressure to improve time-to-market [11].

2.4 Proximity to Market and Customer

By establishing subsidiaries in countries where the company's customers are located, GSD allows it to develop software close to their customers and to increase knowledge of the local market [11]. Creating new jobs can create good will with local customers, possibly resulting in more contracts [15]. Indeed, it may be a business necessity to locate closer to customers in order to expand to other markets. For example, a company that develops software for embedded systems may focus on large manufacturing companies based in China, or a software automotive company may locate part of the development in Germany. Development activities may even be located on the same campus as the organization's large customer. Companies may also look to establishing strategic partnerships to gain access to new markets [16].

3 The 'Unknown' Benefits of GSD

Above, we have highlighted the four 'known' benefits which have been cited as driving forces towards the globalization of software development activities. However, there have been individual reports of additional benefits that may be realized through GSD. Up until now, these benefits have been mostly overlooked. Indeed, and as the label reflects, the benefits covered below are not as obvious as the 'known' benefits mentioned above. While the benefits which are well-known tend to easily affect company policy, we believe that additional and sometimes 'hidden' benefits may offer great potential and indeed contribute in strategic company decisions.

While some of the 'unknown' benefits we identify are applicable at organization level (as is the case with 'known' benefits), they also seem to affect coordination and collaboration within and between GSD software teams as well as the basic software engineering tasks at hand. For the purpose of this paper, we therefore use the categories organization, team and process/task when discussing the identified benefits. By focusing on leveraging the full range of benefits, we argue that companies may reap more rewards from their GSD activities, and that GSD may not need to be seen as only a challenge for software development companies.

3.1 Organizational Benefits

Two of the unknown benefits apply primarily at the organizational level. We refer to these as 'innovation and shared best practice' and 'improved resource allocation'.

3.1.1 Innovation and Shared Best Practices

The global business environment demands and expects innovative, high-quality software that meets its needs [17]. Organizations can take advantage of increased innovation and shared best practice that arises from the collaboration of team members who come from different national and organizational backgrounds [10, 14].

In large complex organizations, decentralized, independent individuals interact in self-organizing ways to create innovative and emergent results [17]. Such organizations base their success on innovation and their innovation capabilities come from talent – from their most brilliant, intelligent and creative engineers. Companies that expand into other countries in order to tap into talent have been termed "knowledge seekers" [18]. Such organizations tend to act somewhat differently compared to organizations that offshore purely for cost reasons [14] and we can see an acknowledgement of this benefit through the action of such companies.

3.1.2 Improved Resource Allocation

As an extension to the benefit of access to large multi-skilled labor pools, the organization can benefit from the influx of new (lower-cost) labor in other countries. It may be beneficial for the organization to reassign the newly-redundant higher-cost resources to other, often more strategic, activities while also avoiding the employee turmoil and backlash associated with workforce reductions [19]. Changes in allocation can adhere to the challenge of replacing isolated expertise and instead create skill-broadening tasks and effective teamwork [10].

3.2 Team Benefits

At the team level, we find three unknown benefits, namely 'improved task modularization', 'reduced coordination cost', and 'increased team autonomy'.

3.2.1 Improved Task Modularization

According to Conway's Law, the structure of the system mirrors the structure of the organization that designed it [20]. In fact, it is the product architecture that should determine the team structure, rather than the other way around [4]. In earlier work we have seen the importance of a separation of concerns when decomposing work into modules [21], and it appears that these principles could be extremely relevant for managing coordination complexity.

The nature of GSD leads teams to splitting their work across feature content into well-defined independent modules [10, 22, 23], without "stepping on each other's toes" [4]. This allows decisions to be made about each component in isolation [20]. Partitioning work tasks horizontally results in each site having responsibility for the whole lifecycle of particular functions/modules, it decreases interdependencies and hence, coordination costs [24]. For example, source code branching enables software development teams to work on source code in parallel, and merging the sections once they have been developed [25].

3.2.2 Reduced Coordination Cost

While we acknowledge that temporal distance can prove to be a challenge for GSD teams, it can also be seen as beneficial in terms of coordination: coordination costs are reduced when team members are not working at the same time [26]. The producer of a unit of work can complete the work during the off-hours of the person who requested that work. In essence, coordination costs are reduced since no direct coordination takes place when two people are not working at the same time.

3.2.3 Increased Team Autonomy

Gumm [27] found that organizational and geographical distribution of 'software development units' imply a certain degree of autonomy for each unit. The study reported that this autonomy allowed for the necessity to maintain the different working cultures of each team. This was viewed as necessary to maintain the quality of the work of a single team even if this in turn required careful synchronization of the single processes.

3.3 Process/Task Benefits

In addition to the organizational and team oriented unknown benefits outlined above, there are three further unknown benefits that apply primarily at the process/task level. We refer to these as 'formal record of communication', 'improved documentation', and 'clearly defined processes'.

3.3.1 Formal Record of Communication

Since asynchronous communication relies on technologies such as e-mail and fax [28, 29], a written communication history is usually left [7, 30]. This provides for increased traceability and accountability [31], and allow for input from diverse stakeholders, irrespective of geographical location [30].

3.3.2 Improved Documentation

DeLone et al. [32] state that distributed teams have an increased focus on documentation in order to aid their communication. Gumm [27] reported this as an advantage, in that documentation is better supported within distributed project settings. Information is documented and distributed electronically rather than discussed face-to-face, which allows for the passing-on of project specific knowledge in distributed settings.

3.3.3 Clearly Defined Processes

Independent of a project's process maturity, the definition and structuring of processes is a challenge [27]. While distributed project settings seem to challenge process maturity, they also seem to support it. Process definitions are compiled more carefully in distributed settings. It was noted that if team members were co-located, much of the processes would probably not be formalized.

4 Conclusions

As recognized in this paper, some benefits of GSD have been widely cited and can be considered ‘known’ to both researchers and practitioners. However, additional benefits are evident and they have been, to some extent, overlooked. In this paper, we have identified these ‘unknown’ benefits in order to provide a synthesis of GSD benefits. This will hopefully lead to a more informed debate on the topic as well as more informed decisions on whether or not to pursue GSD. As can be seen, a majority of the ‘unknown’ benefits that we have identified apply at team and process/task level. This is probably part of the reason for them not being widely acknowledged as driving factors towards GSD. See Table 1 for a summary of our synthesis of the benefits offered by GSD.

We have also pointed out the on-going struggle between reducing the distances of GSD and making the most of the dynamic context of the global situation. For example, we see attempts to reduce coordination costs by effective modularizing work, while at the same time wishing to leverage GSD by sharing innovation and best practice between teams. The debate on this matter up until now has not been informed by a full synthesis of the benefits. Hopefully, the synthesis of benefits of GSD should lead to an even more informed debate on this matter.

Cost-benefit tradeoffs in GSD are still not fully understood [26]. The GSD community has yet to come to a consensus on which benefits are realistic, and whether or not practitioners should aim for the realization of each of them. For example, it is not yet clear to what extent cost savings can and are being realized. Also, follow-the-sun has been dismissed by many, but is still being promoted (see e.g. [33]). Most probably, certain benefits may only be realistic in specific contexts while some benefits may be mutually exclusive. Below, we present both ‘known’ and ‘unknown’ benefits of GSD, all structured according to the categories of (1) organizational, (2) teams, and (3) process/task.

Table 1. The benefits of global software development

Organizational benefits	Team benefits	Process/Task benefits
• Cost savings • Access to large multi-skilled workforces • Reduced time to market • Proximity to market and customer • Innovation and shared best practice • Resource allocation	• Improved task modularization • Reduced coordination cost • Increased team autonomy	• Formal record of communication • Improved documentation • Clearly defined processes

References

1. Ågerfalk, P.J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., Conchúir, E.Ó.: A framework for considering opportunities and threats in distributed software development. In: International Workshop on Distributed Software Development, Paris, France, Austrian Computer Society (2005)
2. Herbsleb, J.D., Klein, H., Olson, G.M., Brunner, H., Olson, J.S., Harding, J.: Object-Oriented Analysis and Design in Software Project Teams. Human-Computer Interaction 10, 249 (1995)
3. Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems. Communications of the ACM 31(11), 1268–1287 (1988)
4. Carmel, E.: Global Software Teams: Collaborating Across Borders and Time Zones, 1st edn. Prentice-Hall, Upper Saddle River (1999)
5. Nicholson, B., Sahay, S.: Some political and cultural issues in the globalisation of software development: case experience from Britain and India. Information and Organization 11(1), 25–43 (2001)
6. Holmström Olsson, H., Conchúir, E.Ó., Ågerfalk, P.J., Fitzgerald, B.: Two-Stage Offshoring: An Investigation of the Irish Bridge. MIS Quarterly 32(2) (2008)
7. Carmel, E., Agarwal, R.: Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Software 18(2), 22–29 (2001)
8. Mercer: China and India: Comparative HR Advantages. September 9 (2005) (accessed February 27, 2006), `http://www.mercer.com/china-indiareport`
9. Grinter, R.E., Herbsleb, J.D., Perry, D.E.: The Geography of Coordination: Dealing with Distance in R&D Work. In: International Conference on Supporting Group Work 1999, pp. 306–315 (1999)
10. Ebert, C., De Neve, P.: Surviving Global Software Development. IEEE Software 18(2), 62–69 (2001)
11. Herbsleb, J.D., Moitra, D.: Guest Editors' Introduction: Global Software Development. IEEE Software 18(2), 16–20 (2001)
12. Damian, D., Lanubile, F., Oppenheimer, H.L.: Addressing the Challenges of Software Industry Globalization: The Workshop on Global Software Development. In: 25th International Conference on Software Engineering, Portland, Oregon, IEEE Computer Society, Los Alamitos (2003)
13. Suzuki, J., Yamamoto, Y.: Leveraging distributed software development. Computer 32(9), 59–65 (1999)
14. Carmel, E., Tija, P.: Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce. Cambridge University Press, Cambridge, United Kingdom (2005)
15. Ebert, C., Parro, C.H., Suttels, R., Kolarczyk, H.: Improving validation activities in a global software development. In: Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada (2001)
16. Karolak, D.: Global software development: managing virtual teams and environments. IEEE Computer Society Press, Los Alamitos, California (1998)
17. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer 34(9), 120–127 (2001)
18. Chung, W., Alcacer, J.: Knowledge sources and foreign investment location in the US. In: Conference of the Academy of International Business, June 2003, Monterrey, California (2003)

19. Weakland, T.: 2005 Global IT Outsourcing Study. DiamondCluster International, Inc. (2005) (accessed July 5, 2006), `http://diamondcluster.com/Ideas/Viewpoint/PDF/DiamondCluster2005OutsourcingStudy.pdf`
20. Herbsleb, J.D., Grinter, R.E.: Splitting the Organization and Integrating the Code: Conway's Law Revisited. In: 21st International Conference on Software Engineering, IEEE Computer Society Press, Los Angeles, California, United States (1999)
21. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Communications of the ACM 15(12), 1053–1058 (1972)
22. Sahay, S.: Global software alliances: the challenge of 'standardization'. Scandinavian Journal of Information Systems 15, 3–21 (2003)
23. Bass, M., Paulish, D.: Global Software Development Process Research at Siemens. In: International Workshop on Global Software Development, May 24, 2004, Edinburgh, Scotland (2004)
24. Battin, R.D., Crocker, R., Kreidler, J., Subramanian, K.: Leveraging Resources in Global Software Development. IEEE Software 18(2), 70–77 (2001)
25. Herbsleb, J.D., Paulish, D.J., Bass, M.: Global Software Development at Siemens: Experience from Nine Projects. In: 27th International Conference on Software Engineering, St. Louis, Missouri, USA, ACM Press, New York (2005)
26. Espinosa, J.A., Carmel, E.: The Effect of Time Separation on Coordination Costs in Global Software Teams: A Dyad Model. In: 37th Hawaiian International Conference on System Sciences. Big Island, Hawaii, IEEE, Los Alamitos (2004)
27. Gumm, D.: Distribution Dimensions in Software Development Projects: A Taxonomy. IEEE Software 23(5), 45–51 (2006)
28. Kiel, L.: Experiences in Distributed Development: A Case Study. In: ICSE International Workshop on Global Software Development, Portland, Oregon, USA (2003)
29. Boland, D., Fitzgerald, B.: Transitioning from a Co-Located to a Globally-Distributed Software Development Team: A Case Study and Analog Devices Inc. In: 3rd International Workshop on Global Software Development, May 24, 2004, Edinburgh, Scotland (2004)
30. Damian, D., Zowghi, D.: The impact of stakeholders geographical distribution on managing requirements in a multi-site organization. In: IEEE Joint International Conference on Requirements Engineering, IEEE Computer Society Press, Los Alamitos (2002)
31. Ågerfalk, P.J.: Investigating actability dimensions: a language/action perspective on criteria for information systems evaluation. Interacting with Computers 16(5), 957–988 (2004)
32. Delone, W., Espinosa, J.A., Lee, G., Carmel, E.: Bridging Global Boundaries for IS Project Success. In: 38th Annual Hawaii International Conference on System Sciences (HICSS 2005) - Track 1, vol. 01, IEEE Computer Society, Los Alamitos (2005)
33. Carmel, E.: Keynote speech. In: International Conference on Global Software Engineering (ICGSE), Munich, Germany, August 27-30 (2007)