

Challenges of Global Software Development

Audris Mockus and James Herbsleb

Bell Laboratories, 263 Shuman Blv., Naperville, IL 60563, U.S.A.

{audris,herbsleb}@research.bell-labs.com

Our main goal is to review the problems, solutions, and open issues in global software development (GSD), in which the software development activities are distributed across multiple sites. First we introduce the reasons why GSD is becoming more and more prevalent. Then we outline the issues created or amplified by GSD and illustrate various effects it has on a software project. Finally, we summarize the literature on existing approaches to reduce the negative effects of GSD, and list key open research questions.

Most major companies develop software products in a globally distributed fashion. For example, in year 2000, 185 of Fortune 500 companies outsourced software development to India alone and the amount of outsourcing grew at a 53% yearly rate according to report by the National Association of Software and Service Companies (NASSCOM 2000). Globally distributed software development creates a number of unique challenges, some of which are investigated in (Karlsson et al., 2000; Ebert et al., 2001; Herbsleb & Grinter, 1999; Carmel, 1999; Karolak, 1998).

The often cited reasons (see, e.g., Carmel, 1999) motivating GSD are:

1. Limited pool of trained workforce.
2. Necessity of getting closer to customers and using locality specific expertise to customize/localize the products.
3. National policy in some countries where the government may be a customer requiring suppliers to locate R&D facility in that country as a condition of sale or a favorable tax treatment.
4. Differences in development cost.
5. Promise of round-the-clock development that could lead to shorter intervals.

Notably, some of these reasons essentially force companies into GSD, while others promise economic benefits.

Unfortunately, there are significant pitfalls in GSD. Some arise from lack of or differences in infrastructure in different development locations, including network connectivity, development environment, test and build labs, and change and version management systems.

Other major sources of problems are interdependencies among work items and difficulties of coordination. The interdependencies may arise if the architecture of the system does not involve relatively independent modules that can be assigned so that they do not span locations. Different sites may, for example, make conflicting implicit assumptions that persist much longer than if the development were co-located (Herbsleb & Grinter 1999). The coordination becomes an issue because of process non-uniformities, e.g., variances in the definition of a "unit test" may cause mismatched expectations and conflict. The slip in common milestones in one site may affect other sites, but is often not communicated early enough. The differences in time zones may lead to more frequent work handoffs as in round-the-clock development.

Finally, potentially the largest source of problems in GSD are issues related to communication across sites. One potentially serious issue is that project participants have backgrounds that are distinct across sites. They are less likely to have participated in same projects, have experience with different process, had different training, come from different cultures, and speak different native languages. In addition, the participants are much less likely to have unplanned contact with other sites due to absence of face-to-face, hallway, and lunch conversations. There are also many fewer opportunities to interact with remote colleagues because the set of people with whom one communicates at a distant site tends to change rapidly over time (Herbsleb et al., 2000). In organizations with rapidly changing environments and unstable projects, informal communication is particularly important (Galbraith, 1977; Kraut & Streeter, 1995). For example, as requirements change, it is hard for the formal mechanisms of communication, such as specification documents, to react quickly enough.

The lack of frequent contact also typically means that participants know much less about who are experts on various subjects at the remote site, often making clarification of a simple question like who understands a particular module or who is responsible for a particular feature into a time consuming exercise. Even if one identifies the right person, it is more difficult to initiate a contact with the remote site. Because of cultural and language differences, messages may seem strange or rude, and are less likely to be answered. It is especially difficult to communicate urgency because those at another site don't know nearly as much about the context, nor are they as likely to see the subtle cues conveyed by tone of voice and facial expressions.

It is much more difficult to communicate one's meaning clearly and precisely across sites because of differences in culture, language, and education. Working hours often have small or nonexistent overlap so the number of ways one can communicate is restricted to asynchronous methods, such as e-mail and voice mail. Sharing documents and applications can be a particular problem, and much time is wasted trying to determine if parties are looking at and talking about the same thing.

Finally, there may be lack of trust and lack of willingness to communicate openly across sites. This may be due to worry about one's job security, i.e., the perception of "outsourcing" when work is transferred from one site to another site. This can manifest itself in a lack of willingness to share expertise, in the fear that it will make one more easily replaceable.

We survey literature to illustrate how these issues play together to slow work. In (Herbsleb et al., 2001) survey data and data from the source code change management system was used to model the extent of delay in a multi-site software development organization, and explore several possible mechanisms for this delay. It also measured site interdependence, differences in same-site and cross-site communication patterns, and analyzed the relationship of these variables to delay. It showed that compared to same-site work, cross-site work takes much longer, and requires more people for work of equal size and complexity. The study in (Ebert et al., 2001) determined the cost effectiveness of training, found that distributed teams of code inspectors are less effective, and found that feature based development teams reduce field defects. Experiences of setting up feature based development teams and daily build schedules in a distributed development environment are reported in (Karlsson et al., 2000). The authors point out a number of difficulties in setting up such process and note significant advantages that it brings, including reduction of lead time and late-stage defects and reduction in design information.

We summarize de-coupling of work and creating a virtual site as two approaches that may reduce or eliminate problems associated with distributed work. De-coupling of work involves choosing appropriate development models and criteria (Grinter et al., 1999), division of the product according to implicit work-item-based modular structure (Mockus & Weiss 2001), minimizing and formalizing work handoffs, implementing appropriate coordination mechanisms, and adapting the development process.

The creation of the virtual site involves tools, practices, and process. The main steps are first to set a common development environment including change, problem, and version tracking, build, test, and project management. Second, provide infrastructure for collaborative sessions, including good speaker phones, meeting etiquette, Netmeeting type software for sharing applications, and video conference facilities. Third, provide presence awareness information through a common calendar and instant messaging, and set up common practices on how to respond to phone and email messages. Fourth, give a 360 degree view of project information through regular updates of project management information, FAQs, team web pages and expertise locators. Finally, establish relationships by supporting travel to project kickoff meetings, establishing common communication etiquette, and provide training at remote sites.

There are three areas where GSD could benefit from empirical research. First, the basic mechanisms by which distance affects a software project are not known or not quantified. For example, why does interval of the distributed projects appear to be longer than interval of a comparable co-located project? Why do the costs of a distributed project not go down according to lower expenses associated with some of the locations involved? While there exist a number of suggestions on how to improve globally distributed software, there is no research investigating their effectiveness. For example, how much should an organization spend on travel, remote training, and tools? What practices and tools provide the best results? Finally, new practices, tools, and processes should target the discovered mechanisms of inefficiency of GSD.

Bibliography

Carmel, E. (1999). Global software teams. Upper Saddle River, NJ: Prentice-Hall.

- Ebert, C., Parro, C. H., & Kolarczyk, H. (2001). Better Validation in a World-Wide Development Environment. In Seventh international symposium on software metrics. London, England: IEEE Computer Society Press. (to appear)
- Galbraith, J. (1977). Organizational design. Reading, MA: Addison-Wesley.
- Grinter, R. E., Herbsleb, J. D., & Perry, D. E. (1999). The geography of coordination: Dealing with distance in R&D work. In Group '99 (pp. 306-315). Phoenix, AZ.
- Herbsleb, J. D. & Grinter, R. E. (1999). Splitting the organization and integrating the code: Conway's law revisited. In International conference on software engineering (pp. 85-95). Los Angeles, CA, USA: ACM Order Number: 592990.
- Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2000) Distance, dependencies, and delay in a global collaboration. In Proceedings, ACM Conference in Computer-Supported Cooperative Work, Philadelphia, PA. (to appear).
- Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2001). An empirical study of global software development: Distance and speed. In 23rd international conference on software engineering. Toronto, Canada. (to appear)
- Karlsson, E. A., Andersson, L. G., & Leion, P. (2000). Daily build and feature development in large distributed projects. In International conference on software engineering (pp. 649-658). Limerick, Ireland: ACM Order Number: 592000.
- Karolak, D. W. (1998). Global software development. Los Alamitos, USA: IEEE Computer Society Press.
- Kraut, R. E. & Streeter., L. A. (1995). Coordination in software development. Communications of the ACM, 38(3), 69-81.
- Mockus, A. & Weiss, D. M. (2001). Globalization by chunking: a quantitative approach. IEEE Software. (To appear)
- NASSCOM. (2000). Annual NASSCOM report. Available at <http://www.nasscom.org/>.