Geographically Distributed Software Development

James D. Herbsleb, Rebecca E. Grinter, and Lawrence Votta Jr.

Bell Labs, Lucent Technologies

Thomas Finholt

Collaboratory for Research on Electronic Work, University of Michigan

ABSTRACT

Geographically distributed software development holds much promise for increasing market penetration and speeding up development cycles. However, it also comes with a set of new challenges for those developing the software, bought about by the distance among colleagues. This paper outlines a new research project underway to explore those issues and their implications for organizing geographically distributed software development efforts. We also describe the approaches we are taking towards providing solutions — in the form of processes and technology — to address the challenges of working remotely.

INTRODUCTION

Geographically distributed development has become a way of life for many divisions of Lucent Technologies, and all indications are that it will be increasingly prevalent in the future. There are at least three compelling motivations.

First, there is increased global demand for telecommunications products and services due to deregulation and innovations such as digital wireless technology. One condition some governments place on selling products and services in their country is that a certain percentage of development resources be located in that country. Companies like Lucent are strongly encouraged to locate significant development resources in these countries.

Second, there is a desire to speed up the development process and make it more efficient. With development locations in several countries widely spread across the globe, it becomes theoretically possible to develop around the clock by handing off work from one location to another. The potential cycle time reductions of such an arrangement are extremely attractive. Mastering distributed development also would create a large pool of resources, rather than many separate single-location pools, allowing resources to be used more efficiently by moving them among projects as needed.

Third, there are historical reasons for the geographic distribution of the development effort. The various components of the system itself have independent development histories. Some pieces were originally built in the United States, while others came from European standards. We believe that this trend is set to continue, especially as companies invest in reuse efforts that encourage them to take what has been done, rather than rebuild.

Despite the necessity, and perhaps even desirability of geographically distributed development, it is extremely difficult to do successfully (see, e.g., [6]). The Lucent Technologies experience has probably been fairly typical, finding that geographic distribution introduces delays, misunderstandings, frustrations, and inefficiency.

In response to this need, we have undertaken a research program in geographically distributed product development with the following goals:

- Create a portfolio of methods and techniques for addressing communication and coordination issues in geographically distributed collaboration.
- Understand the various forms of collaborative software development.
- Develop models that relate methods and techniques to the various forms of collaboration.
- Understand the infrastructure demands.
- Understand the role of cross-cultural communication barriers.

PILOT ORGANIZATION

We have chosen GSM (Global System for Mobile communication) Wireless development as our pilot because the project has several different kinds of collaborations across sites involving various levels of "coupling." [7].

Some of their cross-site collaborations involve fairly clean divisions of labor and well-defined interfaces between the distributed groups. Other sites are tightly coupled; they exchange information frequently and make decisions that require constant synchronization. The GSM sites also span different languages and cultures, which makes the coordination all the more difficult. Finally, the organization is very important to the business and highly motivated to overcome the substantial barriers it has encountered.

By way of a brief background, GSM is the primary standard for cell phones in Europe. It is growing quite rapidly in nearly all areas of the world outside the US and Japan, and now has the largest global "market share" of all the cellular standards. Lucent Technologies currently has only a small share of the GSM market, but the company has made significant investment in GSM development and has aggressive market goals.

Product

The infrastructure of a cellular network consists of three basic components. The base station subsystem (BSS) is located in the field, with one or more antennas (one at the center of each "cell") and handles communication with handsets. Each BSS is connected to a mobile switching center (MSC), the second major component, which is primarily a telephony switch that joins the BSSs' to the public switched telephone network, and often to data networks and even directly to some other MSCs as well. The third component is the operations and maintenance center (OMC) which monitors the performance of the network and plays a central role in maintaining it. Figure 1 shows a simplified version of a GSM wireless network.

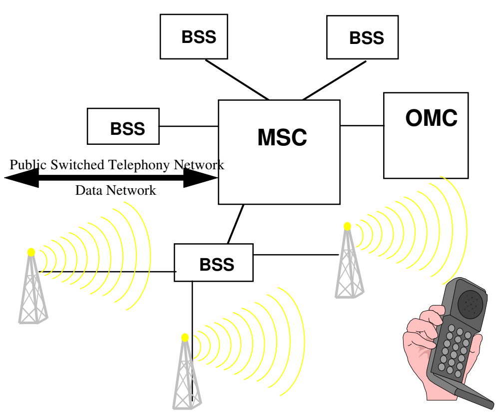

Figure 1. Wireless Architecture

Locations

Figure 2 shows the primary development sites within the GSM organization. This paper focuses primarily on the Swindon and Nuremberg locations. The Swindon site is less than two years old. The personnel came from a variety of Lucent Technologies and non-Lucent sites. The senior managers are ex-patriot Americans. The primary functions carried out at this site are overall GSM management, architecture, project and program management.

The Nuremberg site was only recently acquired by Lucent Technologies, but the personnel have considerable experience working together, albeit for a different company with quite a different culture. Nuremberg is primarily responsible for hardware, manufacturing, modifying and maintaining reused software associated with a previous product, and integration.

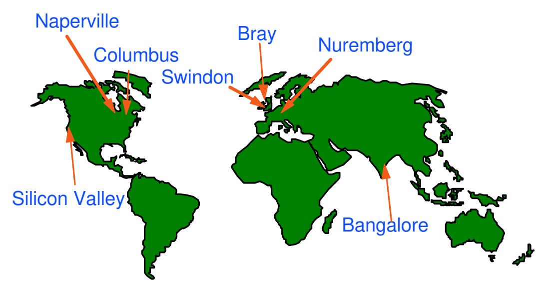

Figure 2. Development locations for GSM Wireless

GEOGRAPHIC DISTRIBUTION: PROBLEMS AND ISSUES

Initially we conducted a series of interviews with GSM personnel in order to identify the perceived difficulties of product development efforts that are distributed across sites. We conducted eleven interviews, spanning technical staff as well as three levels of management, at two sites, Nuremberg and Swindon.

We have decided to focus on Nuremberg and Swindon because the majority of the developers are located at these two sites. Furthermore, we hope that if we can improve the coordination between these two sites that the techniques and tools we develop will transfer to the other GSM sites. The rest of this paper focuses on data from Nuremberg and Swindon.

We asked about each interviewee's background and responsibilities. We then asked the interviewee to describe the kinds of problems they had experienced on this project as a result of geographic distribution. The interviews were conducted by two or three interviewers, who took extensive notes. The interviews were also audiotaped and later transcribed. All of the interviewees had substantial experience in single-site development projects, and were in a good position to recognize the particular difficulties introduced by geographic distribution.

These interviews were followed by a period of residency at one of the sites, Swindon. One researcher spent two weeks on-site to observe the kinds of difficulties that the developers encountered on a day-to-day basis. The observation allowed our initial findings to be confirmed, expanded and revised based on the events that took place. Critically, the researcher was present for the final stages of product release when all the sites were making sure that everything worked together as intended. This researcher also conducted more interviews with other members of staff.

Two problems emerged as very general, and were often mentioned as the source of many of the specific difficulties. Somewhat to our surprise, they turned out to be issues that have received relatively little attention in software engineering: the cultural differences between individuals at the two sites and the absence of informal communication spanning sites. These problems are discussed in the following sections.

Cultural Differences

As mentioned above, workers at the two locations are "culturally different" in at least two senses. The more obvious difference has to do with nationality, i.e., British, German, and American. A difference that may figure just as importantly is corporate culture. As a recently acquired facility, the German site was accustomed to a very different management style. It is often difficult to separate the effects of these two dimensions of cultural difference.

Admittedly, these differences are not specific to software, but seem likely to be typical of product development in general. Yet they would certainly derail any attempt to resolve coordination issues that did not take them into account. These differences are manifested in a number of ways, as described in the following subsections.

Communication style

One of the more obvious problems concerns language. When people from more than one country are present the language spoken by GSM developers is English. The Germans speak and understand English well. However, subtle problems emerge between people who are not familiar with how the Germans use certain words, especially in the context of highly technical discussions. People who reported misunderstandings the most were often those who did not know their German colleagues well and consequently did not perhaps understand the different intentions they had when using certain words. On the reverse side, the Germans sometimes struggled with the speed at which their English-speaking colleagues spoke.

One example of such a misunderstanding was when a German manager said to some English members of his staff that they "should" do something. They immediately dropped what they were doing and completed the new task. Unfortunately, the manager's intention was only that they should "consider" the new task. The difficulty of precise translation of words that often have somewhat different connotations caused a significant misunderstanding.

The quote "two nations divided by a common language" also turned out to have practical ramifications for GSM developers and their management. The management, even those located in Swindon, is virtually all American. Communications between British English and American English speakers sometimes created confusion because some words carry different meanings. For example the word "quite" is used in American English as a positive modifier. In British English "quite" can also be used as a negative modifier, to imply that something could have been better.

Orientation toward process

Obviously, cultural differences run much deeper than language differences. It was easy to see distinct orientations towards following procedures among the British and German developers. These contrasts created tensions among the two sites.

German engineers have a high regard for development processes. They have a fairly detailed defined process and tend to follow it diligently. British engineers do not share this regard for processes, and do not have a comparable defined process. In fact, they are particularly willing to abandon standard processes when situations call for quick delivery.

The balance between following procedure and abandoning it is struck differently at the two sites. From the British point of view it seemed that the Germans would follow the process even when that was going to take too long and cause unnecessary delays. From the German perspective the British developers had little control of their process to begin with, and were far too ready to abandon the process and risk compromising technical quality and reliability.

Orientation toward hierarchy

Workers at the Nuremberg site seem to have a much greater tendency to take a formal approach toward hierarchy, and tend to be more careful in following hierarchy-related protocol. This tends to lead to difficulties as the British sometimes seem disrespectful toward management in the German's eyes, and the Germans sometimes seem to the British to worry too much about exercising discretion in the tasks they carry out. On this particular dimension, the American managers seem to be more similar to workers at the British site.

Informal Communication

Another problem that the developers at both sites talked about was a lack of informal communication. Informal communication such as talk over the coffee machine, at lunch, during chance meetings in the hallway, is surprisingly important for project coordination. This was bought to our attention with comments like:

"Somehow all the tools we've had and all the things we've set up, people still don't talk to each other. That's just the bottom line. They don't talk to each other about the things that matter."

"Chance corridor meetings or lunches is where all the synchronization takes place. The information transfers are not the formal meetings."

"All the little things can grow into big things and they can be stamped out and don't become big things because of the corridor meetings . . ."

Informal communication plays a number of critical roles in development, especially when synchronization of any sort is required. Software's inherent complexity means that interactions between pieces of code owned by different developers frequently occur [1]. These interactions need constant attention, because changes to one piece of code change the way that the other module behaves — a dependency relationship. These dependencies often occur between pieces of code within the same subsystem, and colleagues working in close proximity usually know about these dependencies. Others can span subsystems and one way that people discover these dependencies is through casual conversations with people they meet in the hallway and catch up with.

Informal communication also helps to familiarize individuals with the working styles, orientation towards process and hierarchy, cultural differences of any sort, and helps to solve numerous other problems. In many respects, the absence of informal communication serves to make the other problems that the developers have more intense, because there are no quick and simple ways to resolve the difficulties that they face.

APPROACHES

In this section we outline some of the approaches that we are planning on taking to find out more about the sources of the problems that we have previously described; the contexts in which they occur; their consequences; and above all else techniques and tools that can ameliorate their effects and support distributed collaborative development. We begin with a brief discussion of the iterative prototyping method we intend to employ in all areas.

Prototyping, Iteration, and Data Collection

An important component of this work involves obtaining a deep and rich understanding of the problems that arise in distributed development activities. In order to iterate toward better solutions, we also need to acquire a good understanding of how well various attempted solutions work in practice, and what new problems they create. We will follow an iterative process for devising and evaluating solutions. The four basic steps are

- Conceptualize. We are using a variety of empirical methods to learn about the work environment and distance collaboration needs in depth.
- Prototype. We will recommend or develop solutions (tools, processes, other types of interventions) most likely to bring about significant improvement.
- Trials. As these recommendations are put into practice, we will observe and gather data to determine how well they address the essential problems.
- Modify. Based on these observations, we will make further recommendations to consolidate the advances made in the initial deployment and address identified problems

We believe that this requires a broad selection of empirical methods, including both quantitative and qualitative techniques, as well as strategies for gathering data about depth and breadth.

Qualitative techniques of observation and interviewing have a major role in this project. While the problems of distributed development are increasingly becoming apparent, they remain a source of research investigation. Furthermore, we need to understand the intimacies of how these problems affect this particular environment, these individual developers and the product that they are collaboratively building. Qualitative techniques are essential in order to discover the dynamics of the problems, the solutions that developers have already come up with, and the kinds of techniques and tools that they would like to see and be willing to adopt.

Surveying techniques will also be used for at least two distinct purposes within this project. We will use surveys to initially baseline the organization and to track changes over time. We want to know how prevalent various coordination and communication problems are, and whether the techniques and tools that we implement are having the intended affects or whether something has gone awry. Second, surveys will be used to explore relations among problems, and between problems and other variables of interest such as location, environment, and task. It may be that certain problems tend to occur together, for example, or that certain problems occur in coding, but not in testing. This information will be essential for understanding how to design solutions.

Finally, we will use experiments and quasi-experiments [2] as appropriate. When it is possible either to arrange suitable control conditions or to exploit naturally-occurring opportunities for caparisons, experimentation will generally be the method of choice.

In the following sections, we discuss our initial plans for addressing these problems. They are at various points in the conceptualize-prototype-trials-modify cycle, but all will follow this approach.

Case Study of Success: Test step teams

Our early investigations revealed one significant success in executing a task that spanned geographic locations. The test step teams functioned quite smoothly, reporting almost no problems. At this point, we are not sure why the teams were able to accomplish this feat so effortlessly. We plan to conduct a retrospective case study of these teams, and the processes, tools, management techniques, and the nature of the testing task in order to understand how they coordinated their work so effectively.

The test step teams followed a fairly detailed, well-defined process. One hypothesis about why they were so successful is that this process allowed everyone to understand the current state of testing and how their own tasks fit into the larger whole. If the case study confirms this view, we will explore the possibility of process-based approaches to other specific collaboration problems.

Case Study of a Major Problem: Integration

One problem that we heard much about during our interviews and also saw in practice was difficulties in reassembling the system from the parts. System integration was achieved but not without considerable effort for the organization.

An obvious relationship exists between the process of integration and the high level design. In a sense, the high-level design, which represents the decomposition of the system from its whole into its parts, ought to be the map by which the system is reassembled during integration. That was complicated considerably by the multi-site location of these two elements of the process.

The high-level design team is almost exclusively located at the Swindon site, and during the early phase of the project Swindon becomes the center of all project activity because all the decisions are made there. In contrast the majority of the integration effort happens in Nuremberg. During the final stages of the project all decisions about how to put the pieces back together, what to change to make sure that the system works as intended are made in Nuremberg. Between that initial phase and the final integration phase the work is divided between Swindon, Nuremberg, and the other sites involved in some aspects of the development work. In addition to all the problems created by cross-cultural issues and informal communications, this division also exposes the challenges of a problem of shifting project centers.

Collaboration Technology

A modern software development organization already starts with a wide-base of existing technology. All the developers have access to e-mail, telephones, and the world-wide web. However, there are considerable differences in their infrastructure. Part of the challenge of providing technology to these developers is finding tools that work in heterogeneous environments. This is in addition to all the challenges of adopting groupware technologies generally [5]. These challenges require a sensitivity that we will address by using the four step approach we outlined which helps us understand the problems before we recommend technologies. In other words we are adopting a user-centered approach to the design and implementation of collaborative technologies, where we will test candidate technologies out before placing them into the GSM development environment.

We see three primary roles for technology: linking people to people, people to information, and people to facilities [3]. One that is of particular concern to us is to create opportunities, across geographic distance, for the exchange of information that comes through informal channels when people are co-located. We have already started examining the role that innovative collaboration-facilitation technologies can play within the GSM wireless organizations. Specifically we are considering technologies that allow individuals to meet each other on-line and create virtual opportunities to talk and get to know each other better (see, e.g., [4]).

Tools like the world-wide web have already begun to be used within GSM to link people to information of various kinds. Currently, however the website does not appear to be in full use by everyone. Part of improving the use of the web and other technologies involves determining what kinds of information should be available, how it should appear to the users, as well as improving the interfaces to it. Collaboratories — such as the Upper Atmospheric Research Collaboratory (UARC) — facilitate the sharing of machines and data among geographically distributed scientists [3]. In the context of software development the metaphor of collaboratory creates a new and exciting set of questions about how the data — software, and associated information including documentation, test harnesses, bug reports, and logs — can be visualized, and what common views of that data could be. It is not only the visualizations that support collaborations in a collaboratory but the range of technologies that are integrated together, so that people can see the data and discuss it and edit it at the same time.

We see all these technologies as playing a role alongside the more conventional kinds of meeting support technologies such as shared whiteboards, video-conferencing, and tele-conferencing facilities.

CONCLUSION

We are taking the very first steps in what is designed to be a long-term research program. We have already learned a couple of lessons:

- Many of the worst problems are not strictly technical, but rather have to do with culture, language, and custom.
- Collecting data in order to understand the environment and the problems as fully as possible is and will continue to be critical to our success.
- Informal communication is a crucial coordination mechanisms. Creating opportunities for such communication at a distance is one of the most important challenges.

REFERENCES

1. Brooks Jr., F. P., “No Silver Bullet: Essence and Accidents of Software Engineering,” *IEEE Computer*, vol. 20, pp. 10-19, 1987.
2. Campbell, D. T. and Stanley, J. C., *Experimental and Quasi-Experimental Designs for Research*. Boston, MA: Houghton Mifflin, 1963.
3. Finholt, T. A. and Olson, G. M., “From Laboratories to Collaboratories: A New Organizational Form for Scientific Collaboration,” *Psychological Science*, vol. 8, pp. 28-35, 1997.
4. Fish, R. S., Kraut, R. E., and Root, R. W., “Evaluating Video as a Technology for Informal Communication,” presented at CHI, Monterey, CA, 1993.
5. Grudin, J., “Groupware and Social Dynamics: Eight Challenges for Developers,” *Communications of the ACM*, vol. 37, pp. 92-105, 1994.
6. O'Hara-Devereaux, M. and Johansen, R., *Globalwork: Bridging Distance, Culture, and Time*. San Francisco, CA: Jossey-Bass, 1994.
7. Olson, J. S. and Teasley, S., “Groupware in the Wild: Lessons Learned from a Year of Virtual Collocation,” presented at CSCW, Cambridge, MA, 1996.