Inexperienced and Global Software Teams

Kim Man Lui
The Hong Kong Polytechnic University, Hong Kong

Keith C. C. Chan
The Hong Kong Polytechnic University, Hong Kong

INTRODUCTION

We dream of a way in which software can be quickly or even automatically produced. In the past, we achieved some success in some areas; however, we continue to face an excess of software demand over supply in general. The reason is obvious: Software is needed by not only home and commercial computers, but by almost every kind of electronic device, such as mobile phones, digital diaries, digital cameras, TVs, cars, and so on (Rischpater, 2001; Sapaty, 1999).

Given that the number of qualified programmers cannot be increased drastically and rapidly, software managers in most parts of the world will likely have to live with the human-resources shortage problem for some time (Information Technology of America, 2000). To deal with this shortage, we have to consider forming global software teams in which members are recruited from all over the world and software is developed in a "distributed" manner. Forming such a global software team can have many advantages. In addition to alleviating the problems caused by the scarcity of human resources, programmers on a global team would be free to work without the confines of physical location.

Although the idea of forming a global software team may increase the size of the pool of programmers that can be recruited, quality is a concern. Software managers want programmers who can deliver quality work. But software quality cannot be guaranteed and is hard to justify. Although the managers would prefer to establish a team consisting of experienced programmers, they are in reality faced with the problems of managing inexperienced programmers for a software project, especially in software-developing countries such as China, Poland, South Africa, and so forth.

It is not difficult to see that maintaining a team with a large proportion of inexperienced members significantly reduces running expenses (Figure 1) as there could be a tremendous salary gap between skilled and unskilled developers. Companies that operate with a tight cash flow will normally have an inexperienced software team as they try to minimize costs.

This paper discusses our experiences in running an inexperienced software team and a global software team. We believe that sharing our experience is useful to organizations thinking of exploiting relatively cheaper labor in developing countries such as China, Poland, and South Africa (Sanford, 2003).

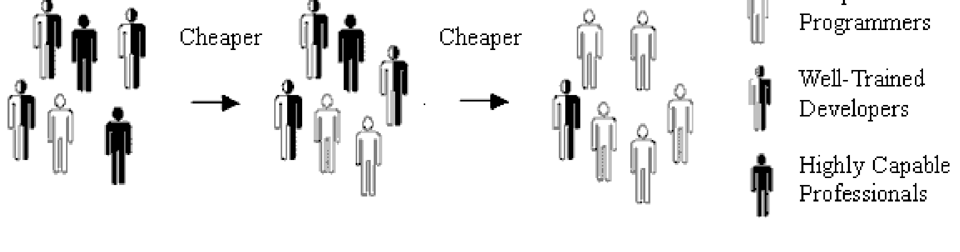
BACKGROUND

This section reviews real cases that have driven the formation of an inexperienced software team and a global software team. The mainstay of managerial decisions for building such teams is financial or environmental.

Discovering Developing Areas: Inexperienced Software Team

It has been a trend for active, rural industrialization, in which manufacturing plants move from more developed regions to less developed ones, to exploit the lower costs

Figure 1. From a professional team to an inexperienced team



of land, labor, and distribution channels (Otsuka, 2001). In order to manage these plants better, management information systems (MISs) need to be developed. Recruiting labor for manufacturing is easy in less developed regions, but recruiting IT professionals for developing an integrated, customized MIS in phases is difficult.

In developing countries, for example, China, the demand for IT professionals in larger cities is so high that it is almost impossible for any manufacturing plant in a rural area to recruit people. Workers in a poor rural area are usually inexperienced. Even though the alternative of employing expatriates might sound reasonable, it is not practical. Instead of in-house development, we might evaluate a third-party solution. The additional expenses incurred in purchasing vendor products, in consultancy services, maintenance, version upgrading, training, traveling, and so forth are expected to be much larger than what can be saved from exploiting cheaper labor. In less developed areas, many programmers do not receive proper training in computing. In addition, the turnover rate is typically high. As long as they have received some training, many workers will seek a job with better career prospects in a more developed city. This results in a vicious cycle in which the project manager always has to work with programmers who are inexperienced. The high turnover rate sometimes is aggravated by resignations without any advance notice. People tender their resignation and leave on the same day. They attempt to keep their current job while seeking any opportunity. Clearly, handing over work is very difficult if not impossible, and the team has to work understaffed constantly.

One may suggest that educating inexperienced people or allocating suitable jobs according to an individual's ability should fulfill the same purposes. However, when the knowledge and experience of staff members is not aligned with the tasks assigned, the learning curve can be steep and long (Amrine, Ritchey, Moodie, & Kmec, 1993). Nevertheless, when a staff member becomes well trained in some less developed regions in China or in a small company in Denmark, for example, his or her determination to look for better job prospects elsewhere will become stronger (Lui & Chan, 2000a; Lui & Chan, 2000b). Training, therefore, does not provide a promising solution in this case. Contradicting what we might expect in well-developed regions, certified professional programs psychologically encourage people to leave a company with little opportunity, or a less developed region, sooner. Some senior managers are disturbed by this phenomenon and say that they are always training another company's staff. The idea of allocating developers according to their skill set is not feasible when all team members are inexperienced. Human resource allocation can therefore be implemented only to a limited extent. Better knowledge management, rather than adopting conventional principles, is required.

Around the Clock: Global Team

A small but ambitious company selling weight-loss and nutritional products, which was headquartered in New York, had a number of small offices of 40 employees in different parts of the world. For each such office, one to two staff members were hired to provide IT support. When the MIS system needed to be modified to meet requirements for local processing, requests for modifications would be sent to the head office. The result was that more resources were required at the head office to provide ongoing support to the branches. Although a larger software team was thus required at the head office, IT staff at the branches might have time to spare. The load-balancing problem got worse when the number of branches increased. The question, naturally, was to decide if it was possible to link people to establish a global software team. The ideal and more productive approach for the above could be a global software team. The team in each site then plays a role more or less as distributed agents following a communication scheme from a coordination agent.

A global software team can even be formed locally if the team is set up in different locations within the same country or in nearby countries or regions. This means that there may not be much difference in time zones and culture. In such situations, the term *multisite* software team can be used more generally to describe a software system developed by teams that are physically separated from each other in different cities of a country or in different countries. Compared with a global team, a multisite team in nearby time zones can be managed with less complexity and fewer challenges. Multisite software teams of this scale, when compared to global teams, have the constraint of relatively limited service hours. A client who sends a request over the Internet around the world normally demands a prompt reply. But if all teams are in the same time zone, meeting the demand immediately outside office hours is not easy. In any case, the management framework required for a global or multisite team should be very similar. To further explore around-the-clock development (see Figure 2) and global development, we realize the intrinsic difference is how synchronization of work in progress proceeds. We concluded that the challenges of managing around-the-clock tasking widely cover managerial and technical problems of non-around-the-clock global software development.

A global team with one site in Asia, one in Europe, and one in North America maximizes time use by working around the clock.