Synching or Sinking: Global Software Outsourcing Relationships

Richard Heeks, *Manchester University*
S. Krishna, *Indian Institute of Management, Bangalore*
Brian Nicholson, *Manchester University*
Sundeep Sahay, *Oslo University*

Clients and software developers need to move their global outsourcing relationships up the value chain to reap greater benefits. Yet such moves bring costs and risks. The authors investigate the strategies that differentiate successful and unsuccessful value chain moves.

G*lobal software outsourcing* is the outsourcing of software development to subcontractors outside the client organization's home country. India is the leading GSO subcontractor, registering average annual growth of more than 40 percent over the last decade and developing nearly US$4 billion in software for foreign clients in FY 1999.[1] Indian firms now develop software for nearly one-third of the Fortune 500.[2]

Advice for potential GSO clients recommends starting small, at home, and with programmers.[3] Many client organizations have followed this advice, putting a toe into the GSO waters through small-scale body shopping—for example, having Indian subcontractor staff come over to the client site to complete a minor, noncritical piece of coding or conversion work. Although this minimizes risk, it also minimizes benefits. In the US, onsite costs of India-related GSO undercut those for hiring US staff by only some 10 to 20 percent, but sending development work offshore to India typically undercuts onsite costs by some 50 percent. Large projects offer a greater potential for savings than small ones. Likewise, the cost savings for hiring Indian analysts or project managers are typically some $1,000 to $2,000 per month greater than those for hiring programmers.[4]

Clients are therefore keen to move up the value chain to increase the cost savings of outsourcing while gaining access to local labor and markets. However, moving up the value chain in this way brings additional costs and risks. GSO of any type imposes communication and coordination costs that local outsourcing does not have.[5] It also imposes risks of total or partial project failure. Clients and developers are therefore seeking routes through the cost/risk minefield to the benefits that higher-value GSO promises. The research reported here investigates these routes. We selected India as the location for the developer half of the relationships to be studied, and North America, Europe, and East Asia as the client half. In the discussion that follows, we use fictitious names to preserve anonymity.

Synching or sinking

In seeking to understand successful strategies in GSO relationships, we were

0740-7459/01/$10.00 © 2001 IEEE

struck by contradictions from prescriptive recommendations. Techniques that worked well for one relationship could cause friction and failure in another. For one client, Gowing, the perceived deference and compliance of Indian developers was a key element in the client–developer relationship's success. However, for another client, Sierra, it was a major problem that contributed to the closure of its Indian operations. Most North American clients found outsourcing of highly structured work to be effective, particularly in reducing transaction costs. But in the Japanese client contracts we studied, outsourcing less structured yet more creative tasks helped build meaningful interactions and relationships.

Therefore, a contingent perspective had to inform our analysis of field data. Classic ideas on contingency and organization relate to the match or mismatch between organizations and their environments.[6] Such ideas are familiar from the business alliance literature,[7] and they also seemed to help explain the dynamics of the GSO relationships we investigated. Successful relationships were those in which a high degree of congruence occurred between developer and client: we call this *synching*. Unsuccessful relationships were those in which a low degree of congruence was achieved between developer and client: we call this *sinking*.

Congruence might exist in relation to any number of contextual dimensions. We developed a dimensional framework on the basis of our initial research and other studies of GSO and of user–developer relationships.[8,9] Using this, synching was more precisely defined as the minimization of gaps between client and subcontractor along the six *Cocpit* dimensions: *c*oordination/control systems, *o*bjectives and values, *c*apabilities, *p*rocesses, *i*nformation, and *t*echnology.

Armed with this framework, we set out to discover the practical realities of synching.

Synching in practice

Those who fail to synch, sink. Their projects run into problems, including outright failures, and they fail to move up the value chain. Take, for example, Pradsoft, a leading Indian software house, which entered into a software development relationship with Ameriten, a large US client. Although there was some sharing of information and overlaps in technology platforms, significant congruence did not develop. The gap arose particularly because of different objectives. Ameriten saw global software outsourcing as a means to cut costs at all costs. Pradsoft, however, saw GSO as a partnership arrangement that should incorporate capacity building and knowledge/technology transfer. Because Ameriten disagreed, transfers did not take place, and the values, processes, and management systems of the two parties significantly differed. Capabilities, too, did not develop as intended. Less than two years after some initial contracts, the relationship was terminated by mutual disagreement, having failed to create synergy or move up the value chain.

We did not find complete congruence in any of the GSO relationships we studied. In the more successful ones, though, synching took place to a significant degree along most of the Cocpit dimensions. In general terms, it was the more congruent relationships that delivered more projects closer to deadline and budget. They also built the preconditions for higher-value outsourcing. In particular, congruence fostered trust between client and developer, and this trust progressed the relationship to larger, more highly skilled projects with more offshore components.

Nevertheless, synching in practice ran into barriers and problems, particularly around certain issues and Cocpit dimensions, as described in the following case studies.

Global and Shiva

Global is a large North American multinational telecommunications company that took a strategic decision in the late 1980s to become involved in global software outsourcing to India.[10] It began to build relationships with a number of developers, primarily with Shiva, one of India's leading software exporters. Involving just 10 Indian staff in 1991, the partnership grew to involve nearly 400 by the late 1990s, mainly based in Mumbai. The relationship deepened to the extent that Shiva worked on some of Global's core technology developments and was designated a full partner lab. This included transfer of ownership of some software products from Global to Shiva, and direct contact between Shiva and some of Global's own customers.

In general terms, it was the more congruent relationships that delivered more projects closer to deadline and budget.

Congruence of this client–developer relationship has also remained subject to factors in the external environment.

Global worked to achieve congruence on all Cocpit dimensions. It succeeded quite well on what can be considered the harder dimensions such as technology and processes. It steadily upgraded its network links until Shiva gained the same high-speed links and access to the corporate WAN as any of Global's own departments. In addition, Global's software development environment was mirrored in India, and the company set aside a dedicated group to ensure that matching technical resources were available at Shiva.

For each project, there was a detailed process of project definition and specification development. This ensured that project methodologies, scope, schedule, and deliverables were unambiguously defined and understood by both parties, helping synch information and processes. Contract negotiations similarly served as a mechanism for building shared understanding, even attempting to address some of the softer components of synching, such as objectives and values.

At both senior and project levels, Global made extensive use not only of the communications facilities available, but also of physical movement to provide for face-to-face meetings. Such meetings proved to be more effective at synching values and informal information, in a way that IT-mediated communication could not. Global also ensured that traffic was two-way, with Global staff visiting Shiva's offices in India as well as having Indian staff come to North America.

Global instituted a comprehensive program of training for Shiva staff to bring capabilities into line with its requirements. This also addressed the process dimension—for example, enabling Indian staff to follow Global's software development methodologies. It covered objectives and values as well by endeavoring to transmit Global's corporate culture and value system. Periods of work for Shiva staff in Global's North American office attempted to do the same thing.

Finally, Global tried to address the coordination/control systems dimension. It supported the introduction of North American HR management systems. Despite barriers, Shiva introduced performance appraisal and career management systems like those operating at Global headquarters.

These techniques built a significant degree of congruence, especially for the capabilities, processes, information, and technology dimensions. Shiva's operations ran quite like those in Global's home country. The result was an average annual growth of more than 15 percent in Shiva–Global business. By the late 1990s, Global was a major Shiva client, accounting for some $8 million of outsourcing contracts per year.

However, Shiva remained an Indian organization based in India, and limits to synching emerged. Perhaps not surprisingly, the limits centered on the soft issues of objectives and values, because they encompass deep sociocultural differences that are not easily erased.

Synching coordination/control systems involved the attempted displacement of Shiva's systems by Global's own. This did not run smoothly because of the cultural underpinnings of those systems. Shiva's systems were deep-rooted and based on a relatively personalized and subjective management culture that had a long tradition. Global's systems were drawn from a culture of objectivity and accountability. Forcing one set of values onto the other was hard, and Shiva's values proved quite resilient. It took enormous efforts before the Shiva project leader would produce a standardized monthly progress report, and Shiva staff refused to participate in Global's employee satisfaction survey. Shiva therefore came to be seen as an outlier in the Global family because its coordination/control systems, objectives, and values could not be pushed fully into synch.

Congruence of this client–developer relationship has also remained subject to factors in the external environment. During the 1990s, Shiva made itself significantly congruent with Global and its expectations and requirements. In the late 1990s, Global's IS strategy took a sudden left turn into Internet-based models and applications. Shiva was in synch with legacy-based models and applications—not with this new world. Synching on every one of the Cocpit dimensions, especially capabilities and processes, began to fall apart. At the time of this writing, the relationship was struggling to get back into synch.

Sierra

Sierra is a small but rapidly expanding UK software house, specializing in high-tech customized projects.[11] By 2000, it had a turnover of roughly $12 million and employed 80 staff worldwide. Sierra began outsourcing development work to India in 1996 using a body-shopping model. Congruence was poor: capabilities were not up to expectations, and—more importantly—Indian staff were outside Sierra's HR systems and company culture, creating factional divisions between Indian and UK staff.

To bring the developers more in synch, Sierra set up an overseas development center in Bangalore in 1998. The center was created on the basis of a rather hastily conducted business plan but with a high level of optimism and expectation about what could be achieved. The main intention was to produce "a little bit of Sierra in India."

Sierra put firm synching strategies into place—it introduced its home processes and coordination/control systems, installed high-speed communications links with videoconferencing functionality, shared information from the UK office over the corporate intranet, and encouraged a youthful and unstructured work environment, like that of its UK headquarters.

Blinded by its enthusiasm, Sierra thought it saw congruence on all Cocpit dimensions and shot the relationship up the value chain. It outsourced whole projects, including virtual liaison with Sierra customers, to the Indian team, envisaging that distance in all its connotations—geographical, cultural, and linguistic—would be invisible.

It deemed some projects successful—especially those that involved members of the Indian development team traveling to the customer site for requirements capture. However, distance could not be made invisible and limits to synching emerged. As might be expected, the videoconferencing link (when not disrupted by bad weather) could not substitute for face-to-face interaction. It failed to transmit the informal information that personal contact provides, creating a barrier to information synching.

As with Global and Shiva, most of the persistent differences clustered along the objectives and values dimension—the dimension, arguably, about which Sierra assumed most and did least to achieve congruence. These differences, in turn, undermined other Cocpit dimensions. One example of cultural dissonance stemming from these differences was rooted in authority. In Sierra UK, authority came more from technical knowledge rather than position. Processes of creative discussion were legitimized, meetings were open and confrontational, and, during problem-solving discussions, junior staff could and did openly contradict more senior staff. In Sierra's Indian office, despite the congruence achieved on some dimensions, the opposite attitude to authority and confrontation prevailed. Another example of cultural dissonance was project leakage: slippage in the amount of time dedicated to a project. In the UK, leakage was typically 2 to 3 percent; in India, it was typically 25 to 30 percent. The problem arose over definitions of leakage. From the UK perspective, it was measured not in terms of deadlines, but in terms of how much time was spent on a task. From the Indian staff's perspective, deadlines mattered but time spent did not, and they felt that UK values were insensitive to Indian conditions. To some degree, coordination/control systems, capabilities, and processes in the Bangalore office were all constrained from synching with the realities or expectations at UK headquarters.

These limits to synching within an overseas development center framework were combined in the late 1990s with an external shock to synching similar to that suffered by Global and Shiva. Sierra moved aggressively into the e-commerce market, where contracts had short life cycles and a strong requirement to understand a lot of specific customer needs.

The limits and shocks threw the client–developer relationship severely out of gear on several dimensions, and, in 1999, Sierra's UK managers closed their Bangalore operation. Having failed to "culturally flood" developers in India, they decided to force synching by repatriating development activities, along with Indian staff, to Britain. This was an attempt to dis-embed that staff from its cultural infrastructure and enforce congruence on the foundational but stubborn objectives and values dimension.

Distance could not be made invisible, and limits to synching emerged.

Table 1

Client–developer relationships, synching, and outcomes.

Relationship	Degree of synching						Outcome
	C	O	C	P	I	T	
Pradsoft–Ameriten	X	XX	✓	X	✓	✓	Insufficient synching leads to relationship failure.
Global–Shiva (pre-shock)	✓	—	✓✓	✓	✓	✓✓	Synching just sufficient for relationship development.
Sierra UK–Bangalore (pre-shock)	—	—	✓	✓	—	✓✓	Synching sufficient for relationship survival but not development.

✓✓: very congruent; ✓: fairly congruent; —: partly congruent; X: slightly congruent; XX: not congruent

Overall results

There are many causes for GSO success or failure. However, from our experience with the cases described earlier and others, synching lays the foundation for higher project success rates in GSO and for higher-value GSO. It is a prerequisite for moves up the trust curve.

Both client and developer managers must understand what synching means in their particular context. They must recognize which dimensions are most important in that context and look for strategies that bring about synching. But they must also recognize synching's serious practical difficulties. For example, while it proved relatively easy to synch technology and capabilities, it proved hard to synch some aspects of information and coordination/control systems and very hard to synch objectives and values (see Table 1).

Limits to synching

Western clients seem to fall too easily for the argument that, in a globalized world, distance, borders, and place no longer matter. The experience of these cases suggests otherwise.

Distance matters, and it interferes with synching because of the difference between GSO's formal/tangible and informal/intangible aspects. Synching strategies in practice have been good at dealing with the former and poor at dealing with the latter, particularly in dealing with three overlapping issues: tacit knowledge, informal information, and culture. In direct terms, the first two are part of the information dimension, and the third is part of the objectives and values dimension. However, their importance derives from the way in which each indirectly affects all Cocpit dimensions other than its own, as Figure 1 shows.

Tacit knowledge. Technologies, specifications, processes, methodologies, skills, objectives, and management systems can transfer from client to developer. But they all have informational components consisting of two parts: the explicit knowledge that can be laid out formally and the tacit knowledge that cannot. Global, for example, went with GSO best practice and used clear, formalized requirements specifications. Although necessary, these were not sufficient because they incorporated a whole set of tacit assumptions and understandings that were not transferred about the nature of the customer, design and programming choices, and working practices.

Clients especially must therefore pay more attention to tacit knowledge. They must recognize its existence within all Cocpit dimensions, identify its content, and look for ways to synch it between themselves and their developers. Techniques might include those described later: use of physical meetings, straddlers, and bridging relationships.

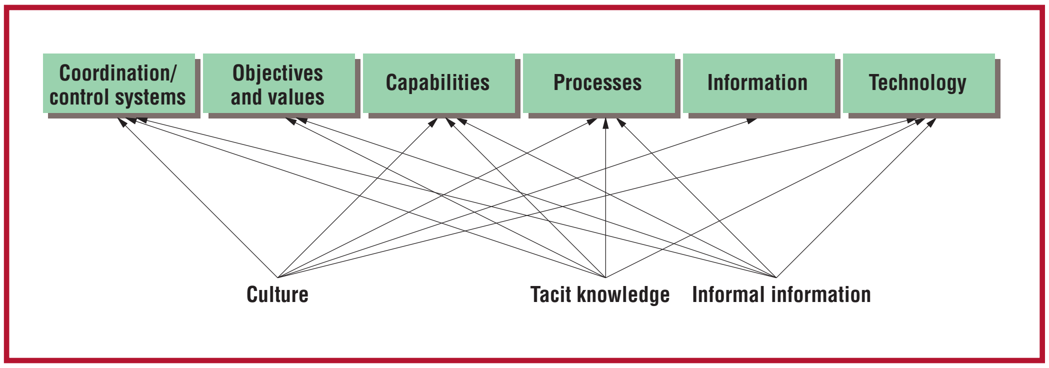


Figure 1. The influence of tacit knowledge, informal information, and culture.

Informal information. Ready transfer of formal procedures also runs into problems because real software development requires constant divergence from formal guidelines and constant improvisation. Informality and improvisation require informal information, which thus remains a critical resource for global software development.

Trying to focus on well-structured, stable projects helped some case study clients push a lot of information exchange into the formal realm that IT-mediated distance can handle relatively well. But informal information cannot be handled this way. Sierra found the cost, fragility, and artificiality of videoconferencing excluded informal conversations and restricted interaction to formal exchange of progress toward milestones. Email, too, only operated at an informal level once participants had physically met and built personal relationships. Travel and direct meetings are therefore a continuous and crucial element in GSO relationships to help fully synch the information dimension.

Culture. Players in this global game still retain cultural values rooted in a particular locale. The overseas development center is a powerful tool for synching and, thus, for raising project success rates and moving up the GSO value chain, but it has its limits. Western processes, systems, capabilities, and so forth can all be imposed. However, some cultural "stains" underpinning these dimensions are hard for this global tide to wash away. Clients must learn to live with this.

Living with limits to synching

How can Western clients retain the benefits of GSO and yet live with the limits to synching? Some pointers were noted earlier, which are specific examples of a more general need to create buffering mechanisms between the distant worlds of client and developer, and bridging mechanisms that allow intangibles to be exchanged between those worlds.

In addition to their synching strategies, clients and developers must therefore also identify the buffering and bridging mechanisms that will help them deal with the limits to synching that still exist within global software outsourcing for some key Cocpit dimensions. Sample mechanisms follow.

Management of expectations. Sierra's overseas development center foundered, in part, because UK expectations were out of synch with Indian realities on several Cocpit dimensions. Those expectations were built on too much media hype and too little cold, hard analysis. As the overly positive expectations crashed down to reality, they left in their wake disillusionment with GSO.

A key, then, to successful GSO is a realistic expectation of what can be achieved given the level of attrition, the limitations of technological infrastructure, the cultural differences, and so forth. Global and Shiva built up a good in-house understanding of what can and cannot be achieved over a long period with a slow-growing expansion and trust curve. This was more successful than Sierra's too-much, too-soon approach.

Using straddlers. Straddlers have one foot in the client's world and one in the developer's world. Global, for example, used some of its India-born managers in GSO relationships. These staff members proved adept at understanding what can and cannot be brought into synch. They managed the dimensional gaps that remained, often acting as a buffer between the Indian developers and senior client managers. Shiva, too, despite high attrition rates, had staff at all levels with significant experience of working in North America who were better able to see things from the client's perspective. These straddlers have also acted as conduits for transfer of tacit knowledge and informal information. Sierra, by contrast (perhaps because it was a smaller and more recent entrant) had no effective straddlers in either client or developer groups.

Building bridging relationships. Straddlers bridge gaps within one individual, but an alternative is to bridge gaps by building strong one-to-one relationships between individual clients and developers. Global and Shiva were active in this, facilitating meetings and other informal contact in which such relationships could develop. They even proactively identified pairs of individuals to bring together. These relationships were a valuable means by which to cope with continuing client–developer mismatch. They created a channel for translating between client and developer and for passing infor-

Players in this global game still retain cultural values rooted in a particular locale.

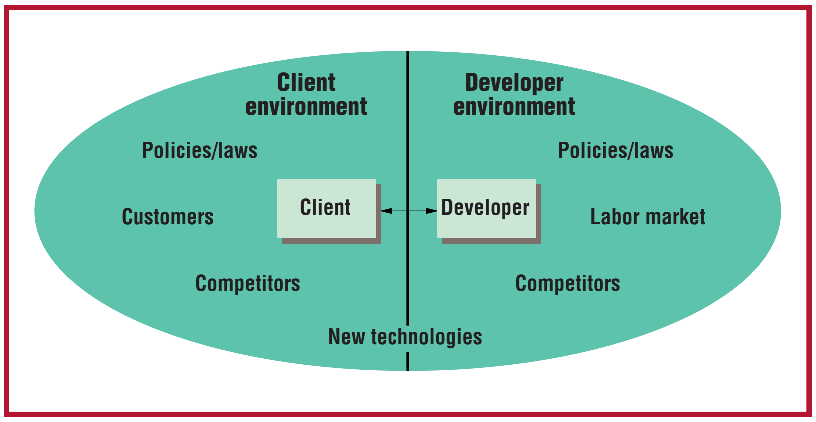


Figure 2. The GSO relationship environment.

mal information and tacit knowledge. They also created the synch of mutual trust and understanding that can help overcome other dimensional differences.

Understanding the synching environment

Both main cases exposed the danger of external shocks to synching, and they are a reminder that the client–developer relationship does not sit in vacuum. Instead, it sits within an environment of which some components are illustrated in Figure 2.

As well as synching with each other, clients and developers also seek to synch with other components of their environment. This creates pressures and tensions for the client–developer relationship, which might be acute (the negative impact of e-business developments) or chronic (the problems of developers in meeting demands of staff in the local labor market).

There is no magic solution to meeting these environmental pressures, especially when they are unexpected. Explicit discussion of the pressures will be one step; diversification of both parties into other relationships will be another.

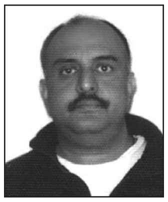
Managers must recognize that there is a seventh dimension to synching: time. Congruence must be actively sustained because, once created, it cannot ensure a permanently synched relationship in a context of continuous environmental change. Clients and developers must therefore continuously reexamine their relationship and proactively move to address emerging mismatches.

About the Authors

Richard Heeks is a senior lecturer in information systems at the Institute for Development Policy and Management, Univ. of Manchester. His research interests focus on the role of IT in governance and in international development. His PhD researched the Indian software industry. Contact him at IDPM, Univ. of Manchester, Precinct Centre, Manchester, M13 9GH, UK; richard.heeks@man.ac.uk.

S. Krishna is a professor at the Indian Institute of Management, Bangalore. His research interests concern global software work arrangements. He holds a PhD in software engineering and chairs IIMB's software enterprise management program, focusing on research and management education in partnership with the local software industry. Contact him at the Indian Inst. of Management, Bannerghatta Rd., Bangalore 560 076, India; skrishna@iimb.ernet.in.

Brian Nicholson is a lecturer in information systems at the School of Accounting and Finance, University of Manchester. His research interests focus on understanding the complexities of software development between the UK and India, which was also the topic of his PhD. Contact him at the School of Accounting and Finance, Univ. of Manchester, Oxford Rd., M13 9PL, UK; brian.nicholson@man.ac.uk.



Sundeep Sahay is an associate professor at the Department of Informatics, University of Oslo. His research interests concern globalization, IT, and work arrangements. Over the past four years, he has been involved in an extensive research program analyzing processes of global software development using distributed teams. Contact him at the Dept. of Informatics, Univ. of Oslo, PO Box 1080, Blindern, N-0316, Norway; sundeeps@ifi.uio.no.

References

1. *Annual Review of the Indian Software Industry*, Nat'l Association of Software and Service Companies, New Delhi, 2000.
2. *The DQ Top 20*, Dataquest, New Delhi, 15 July 1999.
3. F. Warren McFarlan, "Issues in Global Outsourcing," *Global Information Technology and Systems Management*, Ivy League Publishing, Nashua, N.H., 1996.
4. R.B. Heeks, *India's Software Industry*, Sage Publications, New Delhi, 1996.
5. U. Apte, "Global Outsourcing of Information Systems and Processing Services," *The Information Society*, vol. 7, no. 4, Apr. 1990, pp. 287–303.
6. P.R. Lawrence and J.W. Lorsch, *Organization and Environment*, Harvard Univ. Press, Cambridge, Mass., 1967.
7. B. Kogut, "Joint Ventures," *Strategic Management J.*, vol. 9, no. 4, Apr. 1988, pp. 319–332.
8. R.B. Heeks, "Why Information Systems Succeed or Fail," *Proc. Business Information Management Conf.*, CD-ROM, Manchester Metropolitan Univ., Manchester, UK, 1998.
9. E. Carmel, *Global Software Teams*, Prentice Hall, Upper Saddle River, N.J., 1999.
10. S. Sahay and S. Krishna, *Understanding Global Software Outsourcing Arrangements*, tech. report, Center for Software Management, Indian Inst. of Management, Bangalore, 1999.
11. B. Nicholson, S. Sahay, and S. Krishna, *Work Practices and Local Improvisations within Global Software Teams*, working paper, School of Accounting and Finance, Univ. of Manchester, Manchester, UK, 2000.