Development Informatics

Working Paper Series

The Development Informatics working paper series discusses the broad issues surrounding information, knowledge, information systems, and information and communication technologies in the process of socio-economic development

Paper No. 17

Uncertainty and Coordination in Global Software Projects: *A UK/India-Centred Case Study*

SU-YING LAI, RICHARD HEEKS & BRIAN NICHOLSON

2003

ISBN: 1 904143 45 8

Published by: **Institute for Development Policy and Management**
University of Manchester, Precinct Centre, Manchester, M13 9QH, UK
Tel: +44-161-275-2800/2804 Fax: +44-161-273-8829
Email: idpm@man.ac.uk Web: http://idpm.man.ac.uk/

View/Download from: http://idpm.man.ac.uk/publications/wp/di/index.shtml

Educators' Guide from: http://idpm.man.ac.uk/publications/wp/di/educdi.shtml

Table of Contents

Uncertainty and Coordination in Global Software Projects: *A UK/India-Centred Case Study*

Su-Ying Lai, Richard Heeks[1] & Brian Nicholson
University of Manchester, UK

Abstract

Increasing numbers of software development projects are conducted on a global scale that disperses processes to different locations. This paper presents a global software project case study, in which development work was divided between the UK and India. It focuses on two issues: the uncertainties that arise because of differences between project locations, and the coordination activities undertaken in an effort to reduce those uncertainties. The case makes use of the COCPIT framework to identify sources of uncertainty, and a four-way categorisation of coordinative actions. It shows how uncertainty and difference lead to delays, additional costs, and additional management overheads in global projects. It also indicates the limitations of 'remote management' via ICTs. Conclusions are drawn about management of global software projects, including techniques for risk analysis.

A. Introducing the Issues and Case

As a natural consequence of the growth in software development and in globalisation, there has been a significant growth over recent years in global software development projects (Carmel & Agarwal 2002). These are projects that disperse software development processes across national boundaries.

The driving force behind the globalisation of software projects is largely economic. Global dispersal of activities can take advantage of the cost savings and labour availability offered by developing countries (Heeks et al 2001). It also reflects the globalisation of businesses, which find themselves with software requirements and software personnel spread around the globe. All this has been enabled by the global diffusion of information and communication technologies (ICTs).

Globalisation of software projects therefore brings benefits compared with one-country projects. But it also brings problems. Uncertainty – and consequent risk – have been issues plaguing software development since the very first days of software (Sommerville 2001). It is likely that uncertainties will be even greater in global software projects.

Coordination is one way to address uncertainties within software projects (Nidumolu 1995). This can come in a range of mechanisms, from formal standards through to

[1] Corresponding author: IDPM, University of Manchester; richard.heeks@man.ac.uk

informal communication. A global setting poses particular challenges, though, for the selection and implementation of coordination mechanisms.

This paper therefore sets out to investigate the particular uncertainties and particular coordination mechanisms that arise in global software development projects. Its investigation is based on a specific project, centred on a UK—India axis but with client and partner offices dispersed in several other countries. Details of the case project are described next.

The Case Project

"Company A" is a small UK-based firm that provides software development services. It has an annual turnover of roughly US$1m, and has a wholly-owned subsidiary in India. The UK office – with eight staff – is responsible for marketing and client/project management. The India office – with just under twenty staff – undertakes software coding and testing.

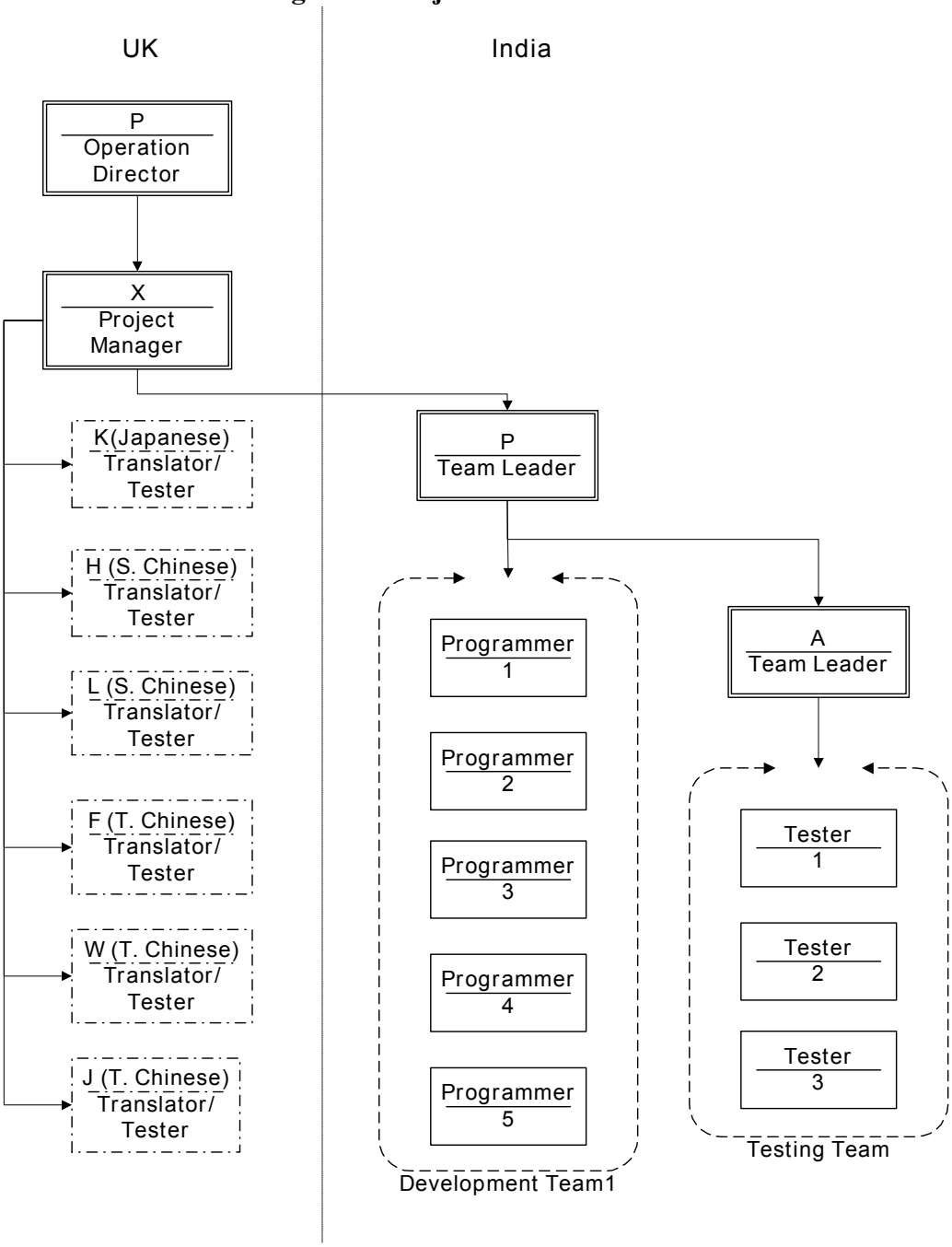
At first sight, the project it undertook from the client seems relatively straightforward. The client was a software product company based in the UK. It wanted the interface of one of its existing software products to be translated into Chinese and Japanese to enable sales into relevant Asian markets.

But the simplicity of the project is deceptive. It had many hidden complexities – perhaps typical of global software projects.

The first complexity arises from the nature of the client and the product. The true owner of the software product – an Internet-based application running on client/server environments – was not the client, but a Taiwanese business partner of client's. It was this business partner that identified the potential demand in Asia for local-language versions of the software. They sent a request for interface localisation to the client's Singapore office which, in turn, relayed the request back to the UK. The client office in the UK then contacted Company A.

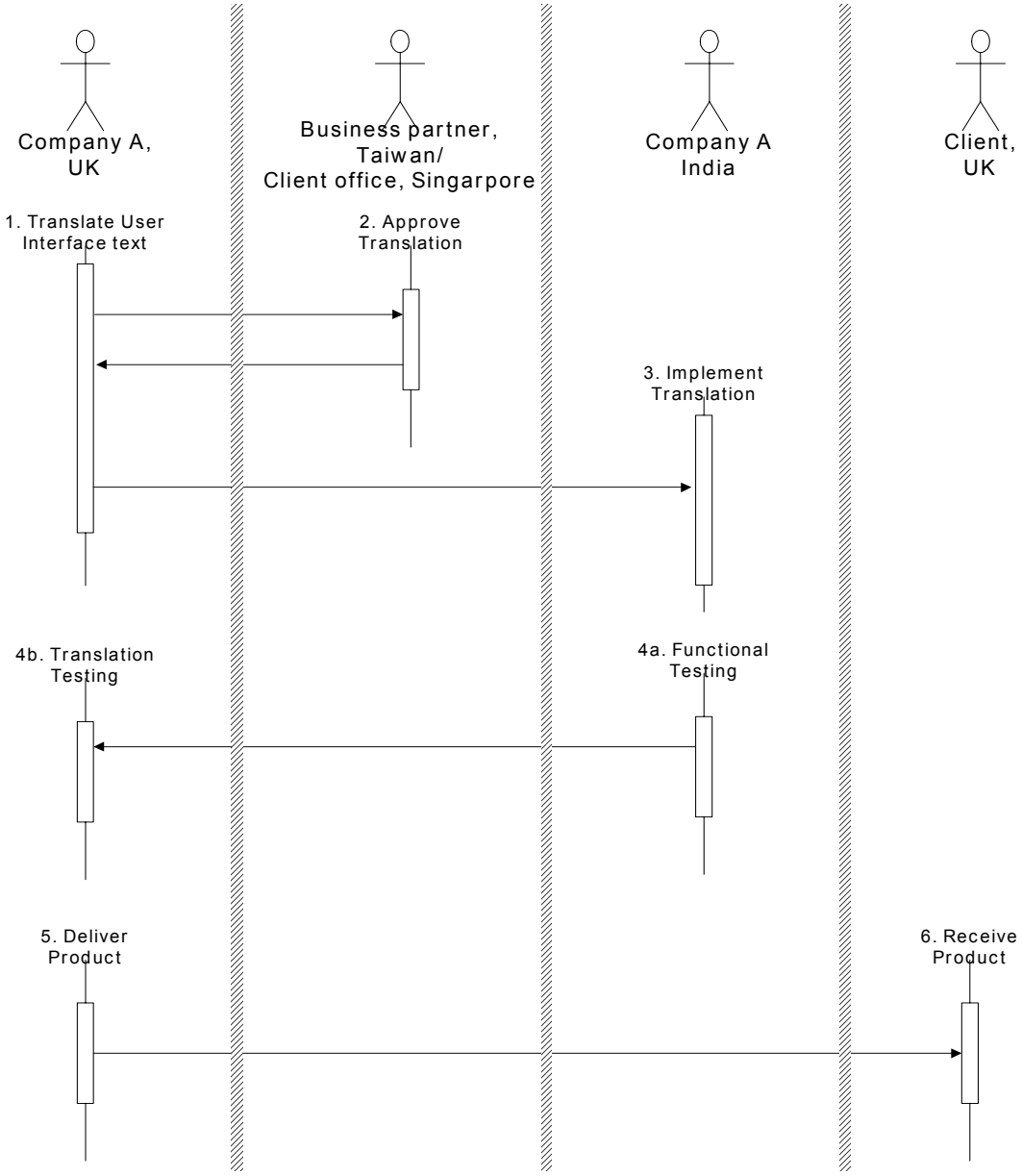
Company A's UK office began the Japanese translation work early in 2002 and hired one Japanese translator in the UK. They began the Chinese translations (into two versions of Chinese – 'Traditional' and 'Simplified') in August 2002 and hired five translators to assist with that work. The UK-based staff translated the interfaces from English into the particular Asian languages. The translated text was then sent over to Company A's Indian office, where it was inserted into the product source code, recompiled, and then tested. The overall project team structure is shown in Figure 1.

Figure 1: Project Team Structure



For the Chinese translation work, the software development process itself was dispersed in four main locations, as summarised in Figure 2.

Figure 2: The Dispersed Software Development Process



Even this simplifies the overall situation, which involved seven sites in four continents (see Table 1 for a summary).

At the beginning of the project, the Company A's UK-based Project Manager – "X" – communicated with the project manager of the client, who was based in their head office in UK. Later in the project, the latter responsibility was reassigned to a manager located in the client's US branch office. Meanwhile, Project Manager X also communicated directly with the business partner in Taiwan and the client's office in Singapore regarding translation approvals.

The developers in India communicated with the client's Australian site directly for technical issues since it was staff there who had developed the original software

product. When the developers in India had problems communicating with the Australian site, Project Manager X would be informed and then he would contact the project manager of the client site in the US to resolve the situation.

Table 1: Inter-Site Communications

	Company A head office, UK	**Company A subsidiary, India**
Client head office, UK	Contract issues Project management issues	-
Client branch office, Singapore	Translation issues	-
Client branch office, US	Project management issues	-
Client branch office, Australia	Technical issues	Technical issues
Client business partner, Taiwan	Translation issues	-

The overall project was more complex still, since a Japanese business partner was involved in the Japanese part of the project, giving advice on translation and market needs.

Data Gathering

Author Lai was employed as one of Company A's translators for six months in 2002/3. During that period, data for this case study was collected through:

- Face-to-face interviews: with the Company A project manager and its business owner.
- Computer-based interviews: two further interviews were conducted via email.
- Participant observation: based on working with the UK team plus members from other sites.
- Informal conversations: with various staff.
- Document analysis: including review of the contract, weekly reports, and project plans.

B. Uncertainty and Coordination in Global Software Development Projects

Uncertainty

Environmental factors introduce uncertainty into software projects. For example, the technological characteristics of software create difficulties (Hughes & Cotterell 2002). The intangibility of software makes it difficult to review progress, thus introducing uncertainty into software project management (Sommerville 2001). Likewise, the flexibility of software is both its best and worst characteristic. "It can be programmed to do almost anything" but "the 'almost everything' characteristic has made it difficult

to plan, monitor, and control software development" because customers or users often think it is easy to change their requirements (Royce 1998:5).

Making the problem worse is the volatile environment that software projects face, in terms of business and technology. Too often, business and technology changes lead to requirement changes and thus increase project uncertainty and risk. In addition, rapid technological changes – in hardware, in application software, in operating systems, in programming languages – make it difficult to plan and manage projects because "lessons learned from one experience may not be transferable to new projects" (Sommerville 2001:73).

Two particular issues can be highlighted as both consequences of environmental uncertainty and as themselves causes of further uncertainty:

- Estimation problems: estimating the time, effort and costs involved in a software project is difficult because of project uncertainties, particularly the 'human factor' (Hughes & Cotterell 2002). Estimation is also hard because many software projects are one-offs that, in some way, differ from all other projects that have come before. Yet estimation uncertainty means it is hard to properly cost, plan and control these projects.
- Requirements uncertainty: meeting the client's requirements is critical to a software project's success – more so than meeting time and budget estimates (Bennatan 1995). Yet ascertaining those requirements is not easy (Redmill 1997). Clients may be unable to articulate their requirements. They may articulate the wrong requirements. Different client groups may disagree over requirements. Their articulation of requirements may be misunderstood by the developers. As a result of this and environmental volatility, requirements may change during a project. This uncertainty leads to conflict, delays, cost overruns, and failure to meet the client's needs.

In summary, these and other forms of uncertainty create problems for software projects. They create efficiency problems: leading projects to run over time and/or over budget. They create effectiveness problems: leading projects to deliver software that is seen by clients to fall short in quality terms.

Uncertainties due to the inherent nature of software and to environmental volatility are found in all software projects. In global software projects, though, there are additional sources of uncertainty because of differences between the different contexts to which project processes are dispersed. Such contexts are often analysed in terms of PEST: the political, economic, socio-cultural, and technological factors within each context. Respective examples would include different legislative frameworks; different skills sets; different cultural attitudes; and different availabilities of telecommunications.

A more project-level model has been introduced by Heeks et al (2001), which uses the COCPIT checklist to identify differences in terms of: coordination systems, objectives and values, capabilities, processes, information and technology. Again, the greater the differences along these dimensions the greater the problems that are seen to arise in global software projects, including the greater the uncertainties..

Particular issues for global software projects include an uncertainty about other members in the project: their objectives, values, and capabilities. This lack of knowledge can knock-on into a lack of trust. In turn, these deficiencies may create miscommunication, misunderstandings, arguments and a lack of team cohesion (Carmel 1999).

Before looking at uncertainty within the global case project, we turn first to look at how to address uncertainty.

Coordination

Action needs to be taken on software projects – including global software projects – to address the high level of uncertainty, and to try to avoid its impact on time/cost overruns and failure to meet client needs. One typical recommended action is 'more coordination'.

Coordination in software projects has been the focus of a number of investigations (e.g. Kraut & Streeter 1995, Montoya-Weiss et al 2001, Andres & Zmud 2002). From these, we can see that coordination works mainly by:

- absolute reductions in uncertainty: removing the variation and volatility that are the sources of uncertainty (e.g. by standardising or formalising procedures), and/or by
- relative reductions in uncertainty: improving the quality and volume of information flowing from sources in order to increase the certainty about those sources (e.g. by holding meetings between different project groups).

In simple terms, we can describe these two facets of coordination as, respectively, control and communication. By addressing the uncertainty of information, they improve decision-making, thus reducing the dangers of inefficient and ineffective project decisions.

A slightly more sophisticated perspective on coordination sees a continuum – rather than dyad – of coordination mechanisms. Sabherwal (2003), for example, divides the continuum into four main categories, described in Table 2. The continuum can be seen as flowing in various ways as one moves down the table:

- from control to communication,
- from formal to informal,
- from absolute reduction in uncertainty to relative reduction in uncertainty,
- from low information-flow requirements to high information-flow requirements (see Barki et al 2001).[2]

[2] There are other perspectives on coordination in software. Nidumolu (1995), for example, distinguishes between vertical coordination (through 'authorised entities' like project managers and steering committees), and horizontal coordination (through peer interactions, e.g. between users and developers). Although both control and communication could, in theory, be delivered either vertically or horizontally, there is a tendency for control to be a vertical mechanism, while horizontal mechanisms tend to be communication-focused.

Table 2: Categories of Coordination Mechanism for Software Projects

Coordination Category	Examples
Coordination by standards	• Compatibility standards • Data dictionaries • Design rules • Error tracking procedures • Modification request procedures
Coordination by plans	• Delivery schedules • Project milestones • Requirements specifications • Sign-offs • Test plans
Coordination by formal mutual adjustment	• Code inspections • Coordination committees • Design review meetings • Hierarchies • Liaison roles • Reporting requirements • Status review meetings
Coordination by informal mutual adjustment	• Co-location • Impromptu communication • Informal meetings • Joint development • Transition teams

These different mechanisms are often seen in a contingent light. Some see contingency in terms of when the techniques are most appropriate (Zmud 1980, Kraut & Streeter 1995):

- More informal, communications-oriented techniques are said to be most suitable when uncertainty is greater: during the requirements analysis phase of a project and/or for larger, more complex types of project
- More formal, control-oriented techniques are said to be most suitable when uncertainty is less: during the design, implementation and testing phases of a project and/or for smaller, less complex types of project.

Others see contingency in terms of the impact of the techniques (Andres & Zmud 2002). They argue that more informal, communications-oriented techniques are more likely to deliver an effective project, with software of a quality that meets client needs. On the other hand, more formal, control-oriented techniques are more likely to deliver an efficient project, that is on time and on budget.

Having reviewed the ways in which uncertainty is addressed, we now move on to look at these issues within the case study project.

C. Uncertainty and Coordination in one Global Software Project

Uncertainty

As noted earlier, the intangibility and flexibility of software can be sources of uncertainty, but this did not emerge strongly as an issue in the case study. Environmental volatility emerged only once, and even then implicitly. In analysing sources of uncertainty in a global software project, we were therefore left with ideas about difference between contexts – described above via the PEST and COCPIT checklists.

The PEST checklist seems helpful in understanding national contextual differences. However, it proved not to work very well when trying to identify or categorise the issues that arose within a specific project since these issues arose from micro-level as much as national-level differences.

We then looked at the COCPIT framework, identifying where differences between project locations relating to each of the six dimensions had led to problems. This did seem more useful but was by no means perfect. As indicated below, differences only clearly emerged on three of the six dimensions (though issues relating to the first dimension – coordination systems – are discussed in the next section), and a couple of issues emerged that could not readily be allocated to any of the dimensions.

Table 3 summarises the differences between project locations that emerged.

Table 3: Summary of Global Software Case Differences

Dimension	Difference
Objectives and values	Differing attitudes to quality appeared during the project. For example, the Japanese partner seemed to have higher quality demands than those that would be expected in the UK. This led to uncertainties in terms of what it meant to meet requirements. The Japanese partner would require small adjustments to an interface that had been considered completed. There were different attitudes to CVs (bio-data). The UK project manager perceived that Indian recruits were not honest in what they put on their CV. Several times, he found Indian staff to have little experience of skills that they had listed on their CVs, and which were of direct relevance to the translation project. This created an irritating uncertainty for the manager about how to truly identify the right person for the right job on the project. It also undermined trust. There were different attitudes to communication and authority. In telephone conversations, the UK project manager found the Indian team leader just saying a simple 'Yes' to whatever was said. The manager was therefore uncertain whether or not he had been properly understood.

	There were different attitudes to work/life balance. The values of Indian staff meant that, if a family member came to visit them, they would take time off from work. They would also take particular religious holidays all together. From their perspective, they were willing to make up the work inputs through overtime at other times. But the UK manager found this all introduced uncertainties into his Anglo-centric methods of project planning. There were different objectives that pertained at different sites. Since there was no documentation for the original product, Company A had to rely on the client's Australian team for key technical support. However, the Australian team was not fully allocated to this project, but had other work that was their main objective. It sometimes took them some days to respond to queries that were sent over from the UK or India.
Capabilities	The Indian software sector has a growing number of skilled workers. Unfortunately, Company A found it difficult to recruit such staff. The reasons were not related to the national context, but were company-specific. Company A's Indian office was in an outlying area of Mumbai (Bombay), a location that raised recruitment barriers because of relative distance from many housing centres, and because of relatively poor transport infrastructure. Being small and seeking to remain cost-competitive, Company A's salary rates were lower than those of many larger firms. This also raised recruitment barriers, meaning most Indian staff were young, inexperienced programmers. This did not appear to have a strong impact on uncertainty, but it did affect the ability of the team to put in place some of the more formal approaches to control that other software projects use. Language was central to the project since the main task was to translate the product interface into different languages. But the UK-based translators did not have English as a first language. Worse, none of the Indian staff spoke Chinese or Japanese. So, for example, it was difficult for them to judge whether or not the Chinese characters they programmed were appearing correctly on screen. This directly added to uncertainty during the testing phase of the project. There was a rather more mundane side to the language issue – the UK project manager found it hard to understand some of the Indian staff when they spoke on the phone, bringing greater uncertainty into the communication process.
Technology	Partly due to cost, but also because the Indian office was in a new location without ready broadband access, the UK and Indian offices used a 64kbps ISDN service, making services such as video-conferencing impractical. Unfortunately, both this link and the Indian power supply were unreliable, especially during the monsoon season, creating further uncertainties for communications.

	There was an unexpected technological problem over operating systems. The main target market for the Chinese-interface product was Taiwan, where Traditional Chinese versions of operating systems – particularly Windows NT4 and Windows 2000 – were the main software platforms. Company A therefore needed these products for its testing environment. Unexpectedly, they were not available in the UK. The company approached the client's sales office in Singapore but found that the NT4 system was no longer available (the one instance in which environmental volatility showed its face). It took more than one month for the sales rep to eventually track down a copy and send it over for use in India and the UK.
Other	Because there were so many personnel dispersed over so many sites, it was sometimes difficult to understand or identify who exactly was responsible for what on the project. This also introduced uncertainties as staff could be unclear who they should be directing particular questions or requests to. The contract stated that the final product should support both Traditional and Simplified Chinese character sets, but it did not specify the formats required. Some formats support only one of the two types of Chinese characters; others can support both. Two months into the project, the UK manager discovered that the original product format-handling was such that it could not support both types of Chinese character at the same time. Yet this was a market need. It had been implicitly understood by the Taiwan partners thanks to their local market knowledge, but it was not made explicit within the requirements and so was not known by Company A staff. The result was that the original product had to be redesigned and rewritten to incorporate the dual-use format.

In sum, there were a significant number of contextual differences that bred uncertainty on the project. What impact did this uncertainty have? The major theme of impact was delay:

- delay as exacting Japanese quality standards were met by making small interface adjustments,
- delay when staff in the Indian office were absent,
- delay while waiting for the Australian team to respond,
- delay as misunderstood phone communications had to be repeated through other channels,
- delay while waiting for staff to be in the office at one of the different global locations,
- delay as files had to be resent,
- delay as particular operating systems were tracked down,
- delay as emails were passed from person to person until they found the responsible recipient, and
- delay while the original product was being redesigned and rewritten.

Delay also meant extra costs particularly because delay either wasted staff time or required additional staff time. Time required, and costs incurred, were thus higher-than-expected.

The other significant impact theme that emerged was a higher-than-expected coordination overhead on the project. This was needed to sort out issues like understanding Japanese quality requirements, placing the right person in the right job, checking translations with competent staff, finding the correct operating system, and problems over the Chinese character set formats.

Coordination: Addressing Uncertainty

As we have seen above, the levels and impacts of uncertainty on this project were significant. What reaction did this trigger in use of coordination techniques? We will investigate this using Sabherwal's framework, described above in Table 2. A summary of findings regarding the more formal techniques is presented in Table 4.

Table 4: Formal Coordination Techniques Used in Case Project

Coordination Category	Usage
By Standards	There was little use of standards on the project in terms of data and procedural rules. The only procedure that came close was that for error tracking, where there was a database in use for identified bugs. In practice, this was only used by the UK translators, rather than by the whole project team, and it was only used as a means for the translators to report bugs to the Indian developers, rather than as the basis for discovery-to-removal workflow tracking. From the Indian side problems were reported only informally, via echat or email. Sometimes the Indian developers would report them via a chain of command, to their team leader who would then report to the UK project manager. At other times, though, developers would contact the UK project manager or even the UK translators directly. There was some level of formal reporting (formal mutual adjustment) in that problems were recorded in a formal weekly report sent from Company A to the client. In practice, though, this was more a document of record rather than a communication device because most problems had already been communicated informally by email or phone.
By Plans	There was use of plans within the project. The contract with the client included a scope document, milestones, a delivery plan, a test plan, and acceptance criteria. These were used as an overall framework guiding the activities of the project. However, this framework could be quite general, leaving actual procedures to be developed relatively flexibly by the UK project manager ("X") and his team.

By Formal Mutual Adjustment	The only regular formal mechanism used in the project was the weekly report. The team leader in India would prepare a weekly report, including information such as 'achievements since last report', 'key issues', 'risks', 'current project task priorities', 'current project activities' and 'progress vis-à-vis project milestones'. Based on this, Project Manager X would send a report to the client project manager. As noted, though, the role and importance of this report was somewhat undercut by informal reporting. At the start of the project, there were also regular project report meetings at the client's UK head office. These were a mix of formal and informal. Project oversight was then transferred to the client's US manager. Meetings were then replaced by occasional phone calls and more frequent use of email, and there was a reduction in the level of formality. There was also a one-off major formal adjustment, which was the renegotiation of the contract. This had to be renegotiated because of the discovery of the Chinese character set problem which meant that the original contract could not be met on either cost or timing. The new contract included an agreement to deliver the product in four phases in order to ship at least something fairly close to the original delivery date.

Formal coordination mechanisms like the plans, reports and contract did play a role in this global software project. However, it appeared that mechanisms for informal mutual adjustment were far more frequently used, and had a much higher profile within the working lives of project members.

There were two main types of mechanism – remote, technology-mediated; and direct, face-to-face. The former are summarised in Table 5, the latter are discussed below.

Table 5: Remote Mechanisms for Project Communication

Tool	Sites	Frequency	Purpose/Content
Email	Company A UK office with Client offices	Needs-based but fairly frequent	Used for discussion and resolution of most management issues
	Company A: UK office with Indian office	Three or four times per day	Used for relatively simple questions (e.g. from developers to translators) or instructions (e.g. telling developers what to do)
Echat	Company A: UK office with Indian office	Two or three times per day	Discussion and resolution of minor technical issues (email used instead when time differences prevented synchronous communication)

	Indian office with Australian client team	Needs-based but infrequent	Obtaining information about technical aspects of the client product
Phone	Company A UK office with Client head office and Taiwan business partner	Needs-based but focused on just one period	Three-way negotiations about new contract (including schedules, milestones and plans) once character set problem was found
	Company A UK office with Taiwan business partner	Two calls	Clarification of Chinese market requirements
	Company A: UK office with Indian office	Two or three times per week at project start, but then less	Discussion of project management issues

Remote communications were used on a daily basis, were vital to the success of the project, and were used as the medium for a continuous series of relatively minor informal adjustments. However, this global project could not work on a totally virtual footing. Face-to-face meetings were an integral and crucial part of project coordination. In all, Project Manager X from the UK felt a need to visit the Indian office on three occasions, staying for about two weeks each time:

- Project start: the main purpose here was knowledge-building, helping each side to know more about the other (thus reducing uncertainties), and helping to develop some level of trust and understanding between the various players that would facilitate later coordination.
- Project mid-point: this visit focused on solving the technical problem related to character sets. The UK project manager had to make direct analytical inputs that could not be done remotely, and in which the Indian team lacked capabilities.
- Project end-point: this visit enabled the UK project manager to exert more direct management control over the Indian team since there was concern about impending delivery dates, and uncertainty as to whether the team was focusing its efforts on the highest priority activities.

In all three cases, visits were deemed necessary because the remote communication channels were deficient. Those channels proved unable to deliver a sufficiently-rich information flow and unable to reduce uncertainties sufficiently far. Where visits were not used – as during the contract renegotiation – the whole process took a considerable time.

These channels – email, echat, phone calls, and visits – were the communication mechanisms by which the informal adjustments could be made. Indeed, the whole project can be seen as a continuous series of flexible improvisations – some minor, some less minor. There was continuous shifting of priorities and timings due to delays. Other examples have also been given above:

- adjusting to emergent Japanese quality requirements,
- repetition of communication where initial attempts failed,

- reallocation of Indian staff to different roles, and
- drawing translators into the testing process.

A related adjustment was the introduction of remote access software, enabling UK translators to test the new interfaces remotely, logging on to the Mumbai-based servers that held the developed code. This improvisation was undertaken because of the problems experienced in transmitting files between India and the UK. Finally, the decision to introduce visits was itself an improvisation.

D. Conclusions

Uncertainty and Coordination on the Case Project

Uncertainty is an important problem for software projects, and this case supports the idea that uncertainty creates significant issues for global software development projects. Ours is neither a comparative nor statistically-significant survey, but the results at least support the notion that contextual differences can be a key source of uncertainty on global software projects. This, in turn, could support the not very controversial idea that uncertainty is likely to be greater on global than on national software projects.

These uncertainties appeared in many forms. There were the expected uncertainties over requirements and estimation. But there were also continuous, micro-level uncertainties about whether messages had been delivered to the right person, had been understood, were going to be answered in time, etc. There were also uncertainties reflecting some lack of trust of the Indian operation by Company A's UK project manager.

As anticipated, these uncertainties were addressed by various coordination mechanisms. There was a strong emphasis on informal remote communication, though punctuated by the odd burst of more formal and/or more direct and/or more control-oriented coordination. The small size of the project and organisations involved is likely to have shaped the nature of coordination. It was also shaped, though, by the limitations of communication channels.

ICTs have been associated with a promise of the 'death of distance', and of virtual working. In this global software project, that promise could not be delivered. Differences in cultural values, differences in capabilities, and weaknesses in technology meant that remote coordination via ICTs could not do enough to minimise uncertainty. Remote coordination via ICTs was vital for 'ticking over', but it could not sustain the whole management process for this global software project.

We saw above that coordination can be seen in a contingent light. This single case provides no evidence about project contingency, but we found little evidence to support the notion of phase contingency. Similar coordination techniques were used throughout the project, with little differentiation between analysis, design, development and testing.

There was some support for the idea that different coordination techniques have different values. The more formal and the more control-oriented techniques – the contract, the weekly reports, the visits – set the broad-brush direction and boundaries: the what and the when of the project. The informal, remote communications – by email, by echat, by phone – supported flexible improvisations within the boundaries: the how of the project.

Finally, the project is a reminder of the two-way relationship between uncertainty and coordination. High levels of uncertainty demand the use of coordination mechanisms. Those mechanisms – if effective – can reduce uncertainty and improve project performance. Yet those mechanisms are themselves liable to being undermined by uncertainty, as seen in the difficulties and delays encountered in project communication.

Reflections on Project Management

Delay was the key theme emerging as an impact of uncertainty on this project. What could a project manager do to address this?

A glib answer would be to allow more time for global software projects, and build this in to the estimation process. This is easy to write, but much harder to implement given the competitive pressures that drive software developer firms to underestimate. Perhaps more realistic would be to devote a bit more thought – as an 'official' project component if possible – to risk and contingencies.

There was no risk analysis undertaken on this project. Even a fairly basic review (e.g. using the COCPIT model – see below) could have helped to identify some potential sources of uncertainty or other difficulty, and could have helped identify mitigating or contingency actions. Contingency planning could have been as simple as asking "What will we do if (when) there are delays on this project?".

This case also supports the idea that managers on global software projects should admit to themselves the limitations of remote management. Face-to-face contact is likely to remain an integral part of these projects that – either officially or unofficially – must be built into project plans. Without this, the necessary trust, knowledge and control cannot be developed.

Project managers also need to identify mechanisms that reduce knowledge gaps between different locations in a global software project. One source is 'straddlers': individuals with a foot on two camps. With limited fanfare, this was a role performed by the translators, who were able to combine knowledge of Asian languages, context and markets with knowledge of UK language and working practices. An alternative is to bridge gaps by building strong one-to-one relationships between individuals in different locations.

The COCPIT Model

The COCPIT model was based on the idea that differences between client and developer context on each of the six dimensions would lead to project problems: the greater the difference, the greater the problem (Heeks et al 2001). We conclude that

this framework is a useful starting point for understanding global software projects. It speaks more to the micro-level project realities of such projects than the PEST framework, and it does guide us towards sources of uncertainty.

On the other hand, COCPIT now emerges with limitations. It covers some key areas of difference, but the six dimensions are not as comprehensive as intended: risk analysis must be alert to other issues. It is also based on the idea of just two locations. It could be particularly helpful in analysing the two-site situation within Company A (even though this is slightly different from the notion of 'client' and 'developer' as originally envisaged). However, we see that real-world cases can be a lot more complex than this. Including Japan, there were eight different project sites involved in this fairly small project, creating up to 28 two-way differences that could be investigated.

We also see a need to unpack the notion of 'difference'. From this case, four separate issues of difference surfaced:

Reality and expectation. In the project, there were 'real' differences: ones that would consistently be identified by most investigative observers. Examples might be the differences in language or software development capabilities between the UK and India, or differences in cultural values. Such differences did create problems for the project, but they did not often create uncertainties. What caused uncertainties was the lack of knowledge about differences. If you know that Indian staff record skills on their CV in a different way to UK workers, that is a hassle but one that can be worked around. If you know that Japanese quality requirements are higher than those in the UK, you can plan for that. What caused the uncertainty-rooted problems was the UK manager not knowing these things – either having no expectation, or having a wrong expectation. Only when the mismatch between expectation and reality was signalled did uncertainty, and problems, start to materialise. Then, gaps in knowledge became apparent, and coordinative actions – particularly communication – had to take place. Other examples include the discoveries about work/life balance and about Chinese character set requirements.

Stability. Differences do create difficulties. But uncertainties and problems appear greater if those differences change during the project. Examples have just been given of changes in the expectation/knowledge difference, including the emergence of the character set issue. But real differences can also change. One example occurred with the transfer of client oversight from the UK to the US office. This created a new set of uncertainties and challenges.

Limitation. To repeat, real differences do – of themselves – create problems for a project. However, particular problems emerged for this global software project where there was not simply difference, but specific limitations imposed by the situation in one project location. For example, differences in technological infrastructure between project locations can create connectivity obstacles. More serious in this case, though, was the low capacity and unreliability of the Indian power and telecommunications infrastructure. The problem does not stem from the difference to UK infrastructure, but from the inherent limitations of low capacity and low reliability.

Impact. As just described, many differences – especially if related to expectation and volatility – can produce uncertainties that are bad for the project. The impact of differences, though, is not just on uncertainty. They can also undermine the coordination (control and communication) activities that seek to address uncertainty. This seems particularly true of real differences (e.g. mismatches in values and capabilities), and of differences so great in one place that they impose limitations (e.g. weak infrastructure).

Therefore, in applying the COCPIT model for risk analysis of global software projects, we have to move beyond a simplistic understanding of difference. We must look at gaps in knowledge about differences (often expressed in terms of expectations) as much as differences themselves.[3] We must take a dynamic as well as a cross-sectional perspective on difference. And we must see where differences are such that factors in one project location will, alone, give rise to problems for the project.

References

Andres, H.P. & Zmud, R.W (2002). 'A contingency approach to software project coordination', *Journal of Management Information Systems*, 18(3), 41-72.

Barki, H., Rivard, S. & Talbot, J. (2001). 'An integrative contingency model of software project risk management', *Journal of Management Information Systems*, 17(4), 37-69.

Bennatan, E.M. (1995). *Software Project Management: A Practitioner's Approach*, 2nd edn., McGraw-Hill, London.

Carmel, E. (1999). *Global Software Teams: Collaborating Across Borders and Time Zones*, Prentice Hall, Upper Saddle River, NJ.

Carmel, E. & Agarwal, R. (2001). 'Tactical approaches for alleviating distance in global software development', *IEEE Software*, 18(2), 22-29.

Heeks, R.B., Krishna, S., Nicholson, B. & Sahay, S. (2001). 'Synching or sinking: global software outsourcing relationships', *IEEE Software*, 18(2), 54-61.

Hughes, B. & Cotterell, M. (2002). *Software Project Management*, 3rd edn., McGraw-Hill, London.

Kraut, R.E. & Streeter, L.A. (1995). 'Coordination in software development', *Communications of the ACM*, 38(3), 69-83.

[3] One way to do this would be to add in a seventh dimension – Knowledge – that looked at gaps in knowledge and expectations about the other dimensions between key stakeholders in different project locations. This, rather neatly, would expand the acronym to COCKPIT.

Montoya-Weiss, M., Massey, A.P. & Song, M. (2001). 'Getting it together: temporal coordination and conflict management in global virtual teams', *Academy of Management Journal*, 44(6), 1251-1262.

Nidumolu, S.R. (1995). 'The effect of coordination and uncertainty on software project performance: residual performance risk as an intervening variable', *Information Systems Research*, 6(3), 191-216.

Redmill, F. (1997). *Software Projects: Evolutionary vs. Big-Bang Delivery*, John Wiley, Chichester, UK.

Royce, W. (1998). *Software Project Management: A Unified Framework*, Addison-Wesley, Boston, MA.

Sabherwal, R. (2003). 'The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives', *Information and Organization*, 13(3), 153-202.

Sommerville, I. (2001). *Software Engineering,* 6th edn., Pearson, Harlow, UK.

Zmud, R.W. (1980). 'Management of large software development efforts', *MIS Quarterly*, 4(2), 44-55.