The Role of Software Processes and Communication in Offshore Software Development

ANANDASIVAM GOPAL, TRIDAS MUKHOPADHYAY, AND MAYURAM S. KRISHNAN

Offshore software development is a new trend in the information technology (IT) outsourcing field, fueled by the globalization of IT and the improvement of telecommunication facilities. Countries such as India, Ireland, and Israel have established a significant presence in this market. In this article, we discuss how software processes affect offshore development projects. We use data from projects in India, and focus on three measures of project performance: effort, elapsed time, and software rework.

The primary motivation behind offshore development is cost. With substantially lower per capita labor costs, customers benefit from moving as much development work offshore as possible. Offshore vendors benefit from consolidating their equipment and communications infrastructure in their home environment, and from favorable governmental policies, tax subsidies, and access to skilled manpower. Without such skilled manpower, as well as a technological infrastructure of machines, software, and high-speed telecommunication links, vendors cannot succeed.

Vendors also need project management techniques to address their geographical separation with clients [6]. Typically, a small team of developers stationed at the client site interfaces with the customer and handles systems integration, installation, and testing [6]. Initial requirements are usually determined at the client site, with more detailed requirement specifications conducted offshore. Next, the project leader and senior designers assemble the core team, and development begins. Once the software is ready, it is shipped to the onsite members, who integrate the components of the system and carry out acceptance testing. The interface between the client and the off-

ANANDASIVAM GOPAL (agopal@andrew.cmu.edu) is a doctoral student at Carnegie Mellon University in Pittsburgh, PA.
TRIDAS MUKHOPADHYAY (tridas@cmu.edu) is Deloitte Consulting Professor of e-Business and Director of the Institute for Electronic Commerce at Carnegie Mellon University in Pittsburgh, PA.
M.S. KRISHNAN (mskrish@umich.edu) is Mary and Michael Hallman e-Business Fellow and Associate Professor of Computer and Information Systems at the University of Michigan, in Ann Arbor, MI.

shore team is managed using a variety of mechanisms such as information requests, open issues resolution, change specifications, and status review video-conferences.

Several issues affect software project management, including communication, project schedules and plans, and personnel [1]. A study of large-scale software development in a major U.S. firm showed the importance of communication in the development process [9]. Communication and coordination mechanisms in offshore development reduce project uncertainty and improve performance. Software processes, including the sequence of tasks that produce desired software attributes, also play an important role. Disciplined software processes in an organization are thought to have a significant payoff in terms of project success. Curtis et al. call for a unified process modeling approach that deals not just with the individual view of each software development component, but with the entire development process, across all levels of the organization [2].

This stream of research culminates in the Software Engineering Institute's Capability Maturity Model (CMM) [12], which we used as the basic framework of our research. The CMM consists of five maturity levels and 18 key process areas (KPAs). Each KPA addresses a set of related goals that must be fulfilled by a set of processes within the organization. KPAs are thought to increase the productivity of software development as they become disciplined and controlled (see sidebar "The Benefits Of Process Improvements and Disciplined Practices"). We collected data on seven KPAs and analyzed their effects on the three performance measures. Our study is the first to examine the impact of KPAs on project rework and elapsed time. We also examined the effect of the contract type on the performance measures. Generally, two kinds of contracts are used in the offshore development context: fixed fee contracts and time-and-materials contracts. Finally, our analysis considers important variables such as prior experience, project size and complexity, and requirements volatility.

Our conceptual model for offshore software development is shown in Figure 1, in which the three project performance measures—effort, elapsed time, and

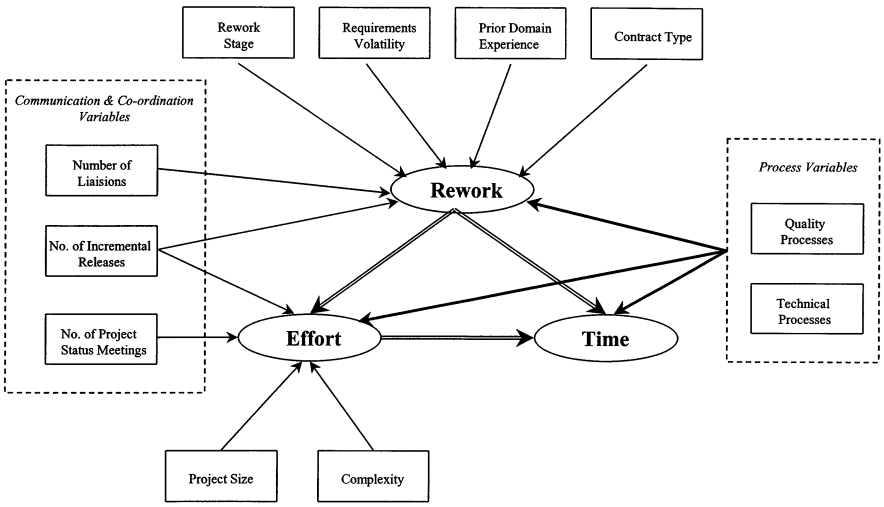

Figure 1. Conceptual model.

rework—appear as three ovals. We postulate that as rework increases, both elapsed time and project effort increase. In addition, we expect elapsed time to increase with total effort. Quality processes and technical processes in the diagram refer to two types of process areas that represent seven KPAs from the CMM. We expect both to affect all three performance measures. We also study three communication and coordination mechanisms in this study: project status meetings, number of incremental releases, and the number of liaisons. The other variables included in this model are: contract type, rework stage, requirements volatility, prior experience, size, and complexity.

We collected data on 34 application software projects from two Indian software firms, both major players in the offshore development area and among the largest software houses in India. Both firms are ISO 9001 certified, and fully Indian owned. Their clients hail from countries ranging from Argentina to Japan, and include computer manufacturers, consumer industries, banks, and financial institutions. Their projects involved implementing client-server applications using fourth generation languages (4GL), and project domains varied from point of sale systems to financial applications for banking institutions. All projects were completed between June 1994 and July 1996. The main instrument of data collection was a questionnaire distributed to the project leaders. The questionnaire included a CMM-based key process area section, previously tested in a U.S.-based software development lab. The variables in our model were:

- *Effort.* This variable was measured in person-days billed to the project.
- *Size*. One firm measured project size in terms of screens and reports, while the other used function points. To get consistent measures of project size, we asked two offshore development experts to judge the project size on a scale of 1 to 5. Their estimates were highly consistent (correlation = 0.92). The average of the two scores was used in our analysis.
- *Elapsed Time*. The project duration in calendar days.
- *Rework*. Data on the percentage of rework during the development life cycle was provided by the project leader, and was cross-checked by the business unit manager. Most projects involve some rework due to changing business rules, user feedback on prototypes, or changing requirements.
- *Complexity*. This variable, based on Boehm [1], captures the complexity of control, computational, data management, and device dependent operations in a project.
- *Number of Incremental Releases*. Vendors often release modules or components during the project, to give the client a feel for the system being developed.
- *Number of Project Status Meetings*. Projects involved weekly or fortnightly project status meetings between the client and the development team, conducted through teleconferencing and video-conferencing facilities.
- *Number of Liaisons*. The number of client organization representatives the development team had to interact with.
- *Contract*. The contract variable is binary, with zero for fixed fee, and one for time-and-materials contract. Since the time-and-materials contract is less restrictive, the vendor is more willing to undertake additional changes to the system.
- *Rework Stage*. This variable determines the development stage at which the need for rework arises: requirements, analysis and design, development and coding, testing, or installation and acceptance testing. Theoretically, changes farther into the life cycle have a greater impact on the amount of rework required.

- *Requirements Volatility.* This variable measures the volatility of requirements through the life cycle on a five point scale. This data was provided by the project leader and was cross-checked by the business unit manager.
- *Prior Experience*. A measure of the number of team programmers who had participated in at least one similar project in the past. The level of uncertainty is expected to lessen as the number of team members with relevant experience increases.
- *CMM KPAs*. We had complete data on seven key process areas: requirements engineering, training, project planning, product engineering, software configuration management, peer reviews, and defect prevention. The score for each KPA was calculated by averaging several questionnaire items. We also subjected the KPA scores to an exploratory factor analysis to find the minimum set of independent linear combinations of responses that can explain as much variance as the original items. The factor analysis procedure provides factor loadings of these individual items on the underlying latent factors. An item heavily associated with a certain latent factor will generally have high loading on that factor. In our study, the seven KPAs loaded on two factors: quality processes and technical processes (see Table 1), which were used in the subsequent analyses. While quality processes refer to quality-related activities of the software development process, technical processes relate to the technical environment of the project.

Figure 1 shows the drivers of the three performance measures. The drivers for rework, for example, are rework stage, volatility, experience, technical process, quality process, liaisons, releases, and contract type. The variables in our models are expected to jointly determine the performance measures. Thus we use a multiplicative model [11]. We convert the multiplicative model into a linear specification by taking logarithms.

Since the data was collected from the same sources and projects, it is possible the error components for the three performance measures are correlated. This additional correlation information between the three error terms can be used to get more efficient estimates of the parameters. Seemingly unrelated regression (SUR) utilizes the correlation between the error terms to get more efficient estimates of the parameters. The estimates from SUR are shown in Table 2.

Table 1. The quality and technical processes, factor loadings.

Key Process Areas	Technical Processes	Quality Processes
Requirements Management	**0.707**	0.010
Software Product Engineering	**0.695**	0.146
Software Configuration Management	**0.791**	0.068
Software Product Planning	**0.790**	0.230
Training Program	0.217	**0.769**
Peer Reviews	0.137	**0.803**
Defect Prevention	0.366	**0.740**

Table 2. SUR parameter estimates for rework, effort, and elapsed time.

Independent Variables	ln(Rework)	ln(Effort)	ln(Elapsed Time)
Intercept	-1.888**	3.294**	-0.923*
ln(Size)	-	2.140**	-
ln(Rework Stage)	0.930**	-	-
ln(Requirements Volatility)	1.635**	-	-
ln(Prior Experience)	-0.356**	-	-
ln(Technical Processes)	-0.516	-0.714*	0.702*
ln(Quality Processes)	-1.380**	0.505*	-0.415
ln(Contract)	0.445*	-	-
ln(Liaisons)	0.760**	-	-
ln(Status Meetings)	-	0.150**	-
ln(Incremental Releases)	-0.327	0.242**	-
ln(Complexity)	-	-0.271	-
ln(Rework)	**-**	-0.048	0.135*
ln(Effort)	**-**	**-**	0.455**
Adjusted R^2	**65.44**	**90.31**	**62.88**

* - $p < 0.10$ ** - $p < 0.05$

A Triangular Relationship

The estimates of the triangular model exhibit excellent properties as evidenced by their respective adjusted R2 values. The R2 statistics indicate the percentage of variance in the performance variable explained by the independent variables. Although rework does not significantly affect effort (in a statistical sense), both rework and effort tend to increase elapsed time. We introduced a binary variable in all three models to distinguish between the data of two organizations but the variable was not significant, and was dropped from further analysis. Our modest sample size (n = 34) is one limitation of this research. Also, some variables are based on subjective estimates, and we assume no significant methodological changes during the two-year study period. Following is a summary of our findings:

Rework. The estimate of the rework variable supports the hypothesis that the stage at which the need for rework arises plays a significant role in determining the level of rework required. The estimate is almost equal to one, indicating that clients who request rework later in the life cycle increase rework considerably. The influence of requirements volatility on rework is even stronger, as predicted. Prior experience of team members in the project domain reduces rework

The effect of technical processes on rework is not statistically significant in this sample, while the estimate of quality processes reduces the level of rework. Quality processes consist of the following KPAs: peer reviews, defect prevention, and training program. The common goal of these KPAs is to detect and prevent bugs and help develop programmer skills. Good training programs and regular peer reviews act as internal quality assurance, helping to weed out errors before the product gets to the client—leading to less rework. Defect prevention plays an active role in quality control, and is instrumental in reducing rework. Deephouse et al. suggested the influence of KPAs on project success are mediated by rework [3]; we see some support of their viewpoint in this model.

Fixed fee contracts, with less flexibility for the vendor, result in less rework. Since the vendor organization must bear the full burden of any cost over-runs, it tends to resist any requests leading to further rework.

The two communication and coordination variables—number of liaisons and incremental releases—play different roles in our model. The number of client site

liaisons tends to increase the rework level. This is expected since more people at the client site must come to a consensus on open issues. On the other hand, incremental releases are negatively associated with rework but this variable is not significant. Incremental releases typically constitute module releases and such releases mitigate some of the risks in the contract. This may lead to less rework.

Effort. As expected, size is an important and significant driver of effort [1]. Quality and technical processes exert opposite effects on effort, an interesting finding. Technical processes reduce effort, whereas quality processes tend to increase effort. Technical processes consist of KPAs related to product planning, product engineering, requirements management, and configuration management. Poor management of any of these areas is likely to result in setbacks and inefficiencies. If configuration management is poor, for example, changes made by different team members will not only generate inconsistencies, but lead to confusion. The quality processes, involving activities such as peer reviews and code walk-throughs, are essential to ensure software quality, but they increase the effort required without directly generating code.

Contrary to our expectations, the two communication and coordination variables—project status meetings and incremental releases—both increase the effort required. During our discussion with project managers, they noted incremental releases often cause clients to want more features added to the system. Such customer requests do not constitute rework, but represent new functionalities for the system. The status meetings variable also increases effort. We believe project status meetings may reduce effort in the beginning by clearing up open issues and tackling bugs early. Beyond a certain level, these meetings may generate new ideas, thereby increasing effort. Product complexity is not significant, possibly because of lack of variance in our sample.

Elapsed Time. As mentioned earlier, effort and rework tend to increase elapsed time. It is interesting that technical processes reduce effort, but increase the elapsed time. The common objective of technical KPAs is to bring the project team together, and thus reduce confusion. However, these KPAs also require a great deal of coordination between project members, which in turn necessitates meetings and discussions about the project plan, requirements, and configuration. Unfortunately, scheduling such meetings may itself cause delays and increase elapsed time. In addition, these activities are not billed to the customer, although they demand additional time. Quality processes seem to reduce elapsed time although this is not significant.

Rework increases the elapsed time of the project, as expected. This finding supports our causal model of offshore software development. This model shows that software processes and other independent variables have varying effects on the three performance measures. The model also shows that communication and coordination in offshore development is multi-faceted, and more complex in its effects on rework and effort. Aspects of this construct operate in different ways, and merit closer examination.

Conclusion

Our results show some support for the triangular relationship between effort, elapsed time, and rework. Requirements volatility and the stage at which rework is required significantly affect the percentage of rework required in a project. Prior experience reduces the rework, whereas the number of liaisons increases rework. Quality-oriented processes significantly reduce rework, and more rework is performed on time-

and-materials contracts as opposed to fixed price contracts. Project size has a significant effect on effort, as expected. Technical processes reduce effort, whereas quality processes tend to increase effort. In light of our finding that more project status meetings and incremental releases lead to increased effort, project managers may benefit from using a balanced set of communication procedures between clients and vendors. In the elapsed time equation, rework, effort, and technical processes all increase the elapsed time of a project.

We believe a significant contribution of our research is to concurrently assess the impact of software processes on three performance measures: effort, rework and elapsed time. Ours is a first attempt at understanding the offshore software development context using empirical data. Also, we believe our finding that communication and coordination between customers and vendors has complex effects on project performance, has important implications for future research on software processes and communication in software development.

References

1. Boehm, B.W. Software *Engineering Economics*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1981.

2. Curtis, B., Kellner, M.I. and Over, J. Process Modeling. *Commun. ACM 35*, 9 (1992), 75–90.

3. Deephouse, C., Mukhopadhyay, T., Goldenson, D.R. and Kellner, M.I. Software processes and project performance. *Journal of Management Information Systems 12*, 3 (1996), 185–203.

4. Dion, R., Process improvement and the corporate balance sheet, *IEEE Software 10*, 4 (July 1993), 28–35.

5. Fox C. and Frakes, W., The Quality Approach: Is it Delivering? *Commun. ACM 40*, 6 (1997), 25–29.

6. Gopalakrishnan, S., Kochikar, V.P. and Yegneshwar, S. The offshore model for software development: the Infosys experience. In *Proceedings of the ACM SIGCPR Conference on The Virtual Workplace: The Impact on Individuals, Organizations and Societies.* (Denver, Colorado). April 1996.

7. Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., Paulk, M., Software quality and the Capability Maturity Model, *Commun. ACM 40*, 6 (1997), 30–40.

8. Humphrey, Watts, Snyder, Terry R. and Willis, Ronald R., Software process improvement at Hughes Aircraft, *IEEE Software 8*, 4 (July 1991), 11–23.

9. Kraut, R.E. and Streeter, L.A. Coordination in large scale software development. *Commun. ACM 38*, 7 (1995), 69–81.

10. Krishnan, M. S., C. H. Kriebel, S. Kekre, and T. Mukhopadhyay, An empirical analysis of cost and conformance quality in software products. *Management Science 46*, 6 (June 2000) 745–759.

11. Mukhopadhyay, T. and Kekre, S. Software effort models for early estimation of process control applications. *IEEE Transactions on Software Engineering 18*, 10 (1992), 915–924.

12. Paulk, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V. *Capability Maturity Model for Software, Version 1.1.* Software Engineering Institute, CMU / SEI-93-TR-24, (Feb. 1993).

The Benefits Of Process Improvements and Disciplined Practices

As defined in the CMM, an effective and mature software process must include interactions among employee skill and morale, tools and methods used in all tasks, and clearly defined metrics and methods of products and processes. Despite awareness of the benefits of process improvements and disciplined practices, software organizations have been slow to adopt such practices. It can be difficult to attribute the cause of project improvements to a specific practice. Also, due to the lack of controlled and rigorous empirical evidence to support the benefits of process improvements, there is some doubt as to their effectiveness. However, this doubt has lessened considerably in recent years with several empirical studies reporting benefits from software process improvements and disciplined practices. Recent evidence on the benefits of software process improvements can be classified into case studies, surveys, and empirical studies comparing several projects. Humphrey et al. [8] conducted a case study reporting significant benefits in terms of risk reduction, budget adherence, and intangible gains such as improvement in employee morale. In another study involving 15 projects between 1988-1992 at Raytheon, Dion [4] reported a $15.1 million reduction in rework costs. A more recent study from Raytheon reported significant reductions in product trouble reports and improvements in productivity [5]. Based on quantitative data from about sixty organizations, and survey responses from multiple case studies in 13 organizations, Herbsleb et al. [7] report improvement of 35% in productivity and 39% reduction in post release defects from software process improvements. In an empirical study using data from 43 projects in a large software organization, Krishnan et al. [10] reports a significant positive effect of CMM KPAs on product defects and life-cycle costs.