A research and education initiative at the MIT Sloan School of Management

A Global Survey of Software Development Practices

Paper 178

June 2003

Michael Cusumano
Alan MacCormack
Chris F. Kemerer
William Crandall

For more information,
please visit our website at **http://ebusiness.mit.edu**
or contact the Center directly at **ebusiness@mit.edu or 617-253-7054**

MITSloan

A Global Survey of Software Development Practices

Michael Cusumano, MIT

Alan MacCormack, Harvard University

Chris F. Kemerer, University of Pittsburgh

William Crandall, Hewlett-Packard

Version 3.1

June 17, 2003

Forthcoming, *IEEE Software*

Abstract

The worldwide boom in the use of information technology has spurred software development efforts around the globe. Our ongoing research program focuses on three aspects of the global development of software: (1) collection of quantitative data regarding current practice and performance in a variety of countries; (2) determination of the adoption of competing models of software development across countries, and (3) analysis of the impact of different development practices on performance.

This paper presents the results of a survey of 104 software development projects primarily centered in India, Japan, Europe and the United States. Practice and performance are contrasted by region. Among other findings it appears from the data that Indian organizations are doing an admirable job of combining conventional best practices, such as specification and review, with more flexible techniques that should enable them to respond more effectively to customer demands. If such a trend is replicated across the broader population, it suggests the Indian software industry is likely to experience continued growth and success in future.

1. Introduction

Several authors have written over the past decade on the wide range of different practices available to software developers [1, 2] as well as possible differences in practices and performance levels around the world [3]. The objective of this paper is to report some early descriptive results from a new global survey of completed software projects that attempts to shed light on international differences in the adoption of different development practices. We expect the findings to be of particular relevance to firms that are considering either a greater use of outsourcing in their software development activities, or the potential advantages of establishing an overseas presence.

2. Prior Global Surveys of Software Development

Two surveys in particular helped inspire the current attempt to survey the use of different software development practices as well as possible performance differences, such as in quality and code productivity, in different regions of the world.

In 1990, Cusumano and Kemerer published a survey of forty projects in the United States and Japan [3]. Their main purpose was to add some quantitative analysis to the discussion over whether practices and performance levels in software development at major firms in Japan were comparable, better, or inferior to major firms in the United States. The results showed that both countries were similar in several areas: types and sizes of products developed, development tools utilized, programming languages and hardware platforms used for development, and the degree of prior work experience. The study revealed that in Japanese projects, more time was spent on product design while American teams spent more time on actual coding. The Japanese projects also exhibited

higher levels of reuse, which was of particular interest due to the fact that the overall sample showed a statistically significant relationship between reuse levels and lines-of-code productivity. However, while Japanese projects were completed with fewer numbers of defects and higher lines-of-code productivity, the differences as compared to U.S. projects were not statistically significant at usual levels, perhaps due to the relatively small sample size.

In a later and broader study published in 1996, Blackburn, Scudder, and Van Wassenhove surveyed forty projects from the United States and Japan, and ninety-eight from Western Europe [4]. Their objective was not to make regional comparisons so much as to identify factors that contributed to the speed of development as well as lines-of-code productivity. Several factors proved to be statistically significant at various levels: for example, the use of prototypes, customer specifications, computer-aided software engineering (CASE) tools, parallel development, recoding, project team management, testing strategies, reuse of code, module size, communication between team members, and quality of software engineers. In particular, the researchers found that spending more time and effort on customer specifications improved both development speed and productivity. The results also indicated that prototyping, better software engineers, a smaller team and less code rework contributed to faster development time. Finally, more time and effort spent on testing and integration had a negative effect on overall development time. Their overall results suggest that early planning and customer specifications are crucial to productivity, while “doing it right the first time” is essential for reducing development time.

3. Motivations for our Research

The motivations for our new program of research were threefold: First, we were interested in how practices and performance levels varied around the world. In particular, one issue that remains unresolved is the accuracy of anecdotal reports on high levels of quality, productivity, and on-time performance with regard to projects from Japan and, more recently, India. With respect to the latter, there has been much speculation that Indian software companies put great emphasis on adopting formal practices and processes that can be used as a signal of quality when bidding for outsourcing contracts (e.g., the Software Engineering Institute's "Capability Maturity Model" Level 5 certification). However, previous studies of this topic have used only small sample sizes [3] or have been based mainly on older data from a small number of individual companies [5, 6, 7].

The second motivation for our study was to ascertain the degree to which different types of development practices associated with particular development "models" are used across a large sample of projects, regardless of location. In particular, we were interested in assessing the penetration of different practices falling across a spectrum ranging from those associated with more traditional "waterfall-style" approaches, which tend to emphasize achieving control and discipline in development, to those that underpin more flexible, iterative models of development, which place a greater emphasis on speed and flexibility in adapting to a set of (potentially uncertain and evolving) customer requirements [8, 9, 10].

Finally, we were interested in contributing to the stream of studies in the field that have examined the impact of different development practices on various dimensions of project performance. In particular, we were interested in contrasting the predictive ability of practices associated with more conventional waterfall style models of development,

with newer styles of development based upon iterative methods, including "synch-and-stabilize" approaches. While previous studies have found support for the predictive ability of these newer practices, they have tended to be limited to a small sample of projects carried out in a specific software context (e.g., Internet software). Our intention was to broaden the analysis of these techniques, in a way that we could examine potential dependencies on other contextual factors such as the type of software (e.g., systems versus applications) and the target hardware platforms (e.g., workstation versus PC).

4. The New Global Survey

4.1 Introduction

To achieve our objectives, we conducted a global survey of software development practices during 2001-2002.[1] This global survey followed a pilot survey of around 30 or so projects from Hewlett-Packard and Agilent.[2] Companies participating in the global survey included Motorola India Electronics, Infosys, Tata, and Patni from India; Hitachi, NEC, IBM Japan, NTT Data, SRA, Matsushita, Omron, Fujji Xerox, and Olympus from Japan; IBM, Hewlett-Packard, Sun Microsystems, Microsoft, Siebel, AT&T, Fidelity, Merrill Lynch, Lockheed Martin, TRW, and Micron Technology from the United States; and Siemens, Business Objects, and Nokia from Europe.

4.2 Survey Metho

regions identified. To initially solicit responses we emailed a worldwide list of industry specialists and researchers and asked them to forward our request for participation to their own colleagues and industry contacts. We also approached other interested parties, such as software associations and research organizations (e.g. the Fraunhofer Institute) and software-focused trade journals (e.g. InformationWeek) to promote participation amongst their members and readers. The survey data were collected through the use of a secure web server hosted at MIT, which, due to its university affiliation, was designed to give respondents greater confidence that their responses would be treated confidentially. In addition, the survey was initially piloted on a sample of US projects from Hewlett-Packard.

Given this general approach, an approach which reflects the difficulty in gaining participation for such surveys on a global basis, our results should be interpreted with caution. To the degree that this sample is not representative of the broader population of projects in each region, then our results may not generalize to these broader populations. For example, as the survey required a significant amount of detailed project data, it may well be the case that projects that were under greater managerial control, i.e., better managed projects, would have a greater likelihood of being included in the sample. Therefore, these results may, for example, represent better, rather than average, project performance.

After removing duplicate responses, responses for which insufficient data was provided and responses for projects that were not yet complete, our sample consisted of 104 projects. Table 1 lists some descriptive data on these projects, broken down by major region of origin.

Table 1: Project Descriptions of the Global Software Process Survey

		India	Japan	USA	Europe & Other	Total
Number of Projects		24	27	31	22	104
Software Type	System Software	7 (29.2%)	5 (18.5%)	4 (12.9%)	4 (18.2%)	20
	Applications Software	4 (16.7%)	4 (14.8%)	7 (22.6%)	5 (22.7%)	20
	Custom or Semi-Custom Software	11 (45.8%)	16 (59.2%)	19 (61.3%)	10 (45.5%)	56
	Embedded Software	2 (8.3%)	2 (7.4%)	1 (3.2%)	3 (13.6%)	8
Level of Reliability	High Reliability	8 (33.3%)	12 (44.4%)	8 (25.8%)	4 (18.2%)	32
	Medium Reliability	14 (58.3%)	14 (51.9%)	20 (64.5%)	18 (81.8%)	66
	Low Reliability	2 (8.3%)	1 (3.7%)	3 (9.7%)	0 (0.0%)	6
Hardware Platform[3]	Mainframe	2 (8.3%)	6 (22.2%)	3 (10.0%)	1 (4.5%)	12
	Workstation	16 (66.7%)	16 (59.2%)	19 (63.3%)	15 (68.2%)	66
	PC	3 (12.5%)	4 (14.8%)	7 (23.3%)	1 (4.5%)	15
	Other	3 (12.5%)	1 (3.7%)	1 (3.3%)	5 (22.7%)	10
Customer Type[4]	Individual	0 (0.0%)	1 (3.7%)	2 (6.7%)	2 (9.1%)	5
	Enterprises	23 (95.8%)	23 (85.2%)	21 (70.0%)	17 (77.3%)	84
	In-House Use	1 (4.2%)	3 (11.1%)	7 (23.3%)	3 (13.6%)	14

5. Regional Differences in Projects

Table 1 shows the differences across the regions by various pre-project descriptors of software project, including Software Type, its required Reliability Level, the target Hardware Platform, and the main Customer Type. The 104 projects are relatively evenly

[3] One project did not provide data on hardware platform.

[4] One project did not provide data on customer type.

divided across India, Japan, the United States, and the remainder, which include primarily European projects[5]. The Indian projects show a greater proportion of system software projects, whereas the majority of Japanese and US projects were self-described as Custom or Semi-Custom software. Europe claimed the highest percentage of Embedded software projects.

In terms of required reliability, few respondents identified their project as "Low", as might be expected in such a survey based on self-reported data. Japanese projects showed the relatively greatest difference between High and Low reliability.

Hardware platforms are primarily represented by workstation projects across the survey. Japanese projects reported the relatively highest percentage of Mainframe projects, the US the relatively highest percentage of PC projects, and Europe the highest percentage of the handful of projects that were not primarily classified as one of the above platforms.

The vast majority of projects were described as primarily for Enterprise use. The US sample, however, did include relatively more projects for In-house use, which is consistent with the high percentage of custom projects in the US sample.

6. Regional Differences in Practices

Our survey first asked about conventional practices that follow more of a waterfall-style model and relied on tools and techniques once popular among large-scale software systems developers such as IBM and the Japanese computer manufacturers. For example, how many projects wrote architectural and functional specifications as well as developed detailed designs before coding? How many used code-generation tools? And how many went

[5] This column will simply be referred to as "Europe" in the remainder of the paper although readers should understand that it includes other countries not in Europe, e.g. Israel. In addition, in the discussion that follows all of the comments reflect the data as presented in the tables, and do not reflect any statistical analysis. Therefore, any references to, for example, "significant differences" should be interpreted in the informal, rather than the statistical sense of the word.

through formal design and code reviews? Then we asked about a set of newer techniques geared to making projects more flexible. For example, Cusumano and Selby describe a model referred to as "synch-and-stabilize" in which projects are broken down into multiple "milestones" or subcycles, each one geared to delivering only a subset of a product's final functionality [9]. At the end of each of these milestones is a period where the team "stabilizes" or debugs the code under development and may also choose to release an early beta version for feedback from selected customers [8, 9, 10, 14]. At a more micro-level, the work of developers is "synchronized" by the use of techniques such as daily or weekly builds (and tests) of the code, to ensure that problematic interactions among components are surfaced and corrected at the earliest opportunity. These techniques are often used in conjunction with other approaches for gaining early feedback on a design, for example, the pairing of developers and testers for the purposes of checking code prior to submission. We therefore report data on all of these more flexible practices; how many projects divided development into subcycles or milestones, used early beta tests, paired programmers with testers, followed daily builds, and conducted detailed regression tests (as opposed to simple compile and link tests) on each build? Table 2 contains descriptive data on these measures for our sample of 104 projects.

Table 2: Process Data from Global Software Process Survey

		India	Japan	USA	Europe&Other	Total
Projects		24	27	31	22	104
Arch. specs	%Yes	83.3	70.4	54.8	72.7	69.2
Functional specs	%Yes	95.8	92.6	74.2	81.8	85.6
Detailed designs	%Yes	100	85.2	32.3	68.2	69.2
Code generation	%Yes	62.5	40.7	51.6	54.5	51.9
Design reviews	%Yes	100	100	77.4	77.3	88.5
Code reviews	%Yes	95.8	74.1	71.0	81.8	79.8
Subcycles	%Yes	79.2	44.4	54.8	86.4	64.4
Beta	% >=1	66.7	66.7	77.4	81.8	73.1
Pair Tester	%Yes	54.2	44.4	35.5	31.8	41.3
Pair Programming	%Yes	58.3	22.2	35.5	27.2	35.3
Daily builds	%Beginning	16.7	22.2	35.5	9.1	22.1
	%Middle	12.5	25.9	29.0	27.3	24
	%End	29.2	37	35.5	40.9	35.6
Regression test on each build	%Yes	91.7	96.3	71.0	77.3	83.7

In terms of practices, most of the sample used architectural, functional, and design specification documents, rather than just write code with minimal planning and documentation. These conventionally well-regarded practices were especially popular in India, Japan, and Europe. The major difference was in the U.S., where the sample indicates that specifications are used less frequently across the board. This is most striking with regard to detailed design specifications, which were only reported to be used in 32% of U.S. projects (in contrast to India, where 100% of the Indian projects reported using such a

document). Interestingly, Cusumano and Selby observed a decade ago that Microsoft programmers in general did not write detailed designs, but went straight from a functional specification to coding in order to save time and not waste effort writing specs for features that teams might later delete [9, 14]. Our results suggest that this may be becoming a more widely followed practice in the United States. With regard to other more conventional practices, design and code reviews were also used extensively throughout the sample. In fact, 100% of Indian projects reported doing design reviews, and all but one reported code reviews. This suggests that there is some truth to the recent speculation that Indian software companies are increasingly adopting more formal project management techniques.

With regard to the newer, more flexible development practices, we find these were also popular around the world, with some variations. Over 64% of projects broke projects down into subcycles, for example, though these were more common in the Indian and European samples than in Japan or even the United States. Firms that did not use subcycles, by our definition, followed a more conventional waterfall process. Less than half of the Japanese projects used subcycles, thereby indicating that in this region, a waterfall process is still a popular choice. Almost three-quarters of all projects made use of early beta releases, which have become a useful tool for testing and obtaining user feedback, especially since the arrival of the World Wide Web [8, 10, 15]. There was no clear difference between regions however, in the use of this practice.

At a micro-level, there are also some interesting observations. Daily builds were used at some point in development by only one third of the sample – a surprising statistic given the publicity paid this technique over recent years. More interestingly however, is the pattern of usage noted over a project's life. In particular, U.S., projects tended to decide whether to

adopt daily builds or not, and stick to this strategy *throughout* a project, whereas European projects (and Indian/Japanese projects to a lesser extent) tended to vary the use of this practice over a project's life. Specifically, early in a project, daily builds were not used much at all, but in the later stages, this became much more common. With regard to running regression tests on builds, we see that the number of projects doing this was fairly high – over 80% - with the highest concentrations of this practice coming in Japan and India. Finally, over 40% of the projects surveyed paired testers with developers, and nearly as many reported using paired programmer techniques. Again, these practices appeared to be especially popular in India.

7. Regional Differences in Performance

7.1 Introduction

Project performance is difficult to measure and even more difficult to compare regionally from such a small sample, but we did collect traditional software development measures of performance in order to further characterize our sample. These were the output per programmer-month of effort in terms of lines-of-code, and the defect rate of the product in terms of defects reported per 1000 lines of code in the 12 months after delivery to customers (reported in Table 3).

Table 3: Performance Data from Global Software Process Survey

		India	Japan	USA	Europe&Other	Total
Projects		**24**	**27**	**31**	**22**	**104**
Output[6]	Median	209	469	270	436	374
Defects[7]	Median	.033	.005	.030	.050	.030

[6] Output = New Line Of Code / (average staff * Calendar duration).

[7] Defects = Number of defects reported in the 12 months after implementation / KSLOC. (This ratio was adjusted for projects which had fewer than 12 months of data.)

We report median levels of performance here to avoid the impact that outliers have on sample means. Note that the performance differences observed between regions are likely due in part to the differing project types, underlying hardware platforms, coding styles, customer types and reliability requirements. The numbers are therefore intended to be descriptive only of the data in this sample and not as the basis for projecting the performance of other projects that might be run in future within each region.

Based on the data in our sample, Japanese projects achieved the lowest defect levels (median of 0.005). The Indian projects (0.033) and U.S. projects (0.030) were quite similar to each other, but considerably higher than the Japanese. The European projects had the relatively highest median defect rate of 0.050, but this rate is similar to the U.S. and Indian levels. In terms of lines of code delivered per programmer per month[8] – an admittedly limited measure of output in this context – the Japanese and European projects ranked at the top. They had a median output level of about 450 lines of code per programmer month, unadjusted for programming language or type of project. This was approximately twice the level of the Indian and U.S. projects in this sample. The same caveats about differing project characteristics cited above for the defect data apply to the output data as well.

7.2 Relation to other studies

Various small-sample studies from the late 1980s and early 1990s also found higher levels of code output and fewer defects from Japanese software projects as compared to U.S. projects, so the Japanese results may not be seen as surprising. U.S. programmers often

[8] SLOC/effort is a measure with limited utility when used to make comparisons across organizations, as is done here. It has been used effectively, however, when comparing projects from within a single organization which greatly reduces possibly unaccounted for sources of variation [17, 18].

have different objectives and development styles. They tend to emphasize shorter or more innovative programs, and spend more time in optimizing code, which ultimately reduces the number of lines of code, while simultaneously increasing effort. The Indian organizations often have a significant fraction of U.S. clients, and may well have adopted a U.S.-type programming style, although we expected defect levels to be closer to the Japanese, given the emphasis in the literature of Indian companies on the adoption of more formal practices (e.g., achievement of high SEI CMM levels) that have historically been associated with improving software reliability.

8. Links Between Practices and Performance

The work that seeks to connect specific process choices with particular performance attributes in this broad sample is still ongoing [19]. However, researchers on software engineering over the past two decades should be interested in the findings that emerged from the pilot data for this survey, which was gathered at Hewlett-Packard and Agilent and is forthcoming in *IEEE Software* [13].

First, we found that developers appeared to be more productive in terms of code output when they have a more complete functional specification before starting to write code. Second, having more complete designs before coding appeared to correlate with a lower level of defects. These results make intuitive sense and have led many software managers to insist on having complete specifications before people start writing code – as is advocated in a waterfall process. Programmers can be more productive in a technical sense if they make fewer changes during a project and thus have less rework to do. They also have less chance of introducing errors if they make fewer design and code changes.

Yet, in a business sense, locking a project into a particular design early on may not produce the best product for a customer in a changing market. We also found that use of early betas and prototypes – opportunities for customers to provide early feedback on the design – was associated with higher output rates and lower defects, probably because projects were able to make early adjustments to this feedback (as opposed to finding out that some things were wrong later in a cycle). Furthermore, running regression or integration tests with each build and conducting design reviews (both mechanisms for gaining early feedback on a design) were associated with lower levels of defects.

Importantly, our analysis showed that when we captured the impact of all these various practices in a model predicting performance, the presumed disadvantages that stem from not having a complete specification disappear. That is, adopting practices associated with a more flexible process (i.e., those geared to generating early feedback on a product's performance) appears to compensate for the potential trade-offs arising from an incomplete specification. In a sense, these practices may be seen as providing an alternative mechanism for generating the type of information that a specification typically communicates. Our findings help explain why early research on software development may have concluded that a waterfall style process led to improved performance; they may not have captured data on the use of other (more flexible) practices that were actually better predictors of performance. And they also help us understand that in selecting a development model, we should be careful not to think that we can "cherry pick" only those practices that look most appealing. On the contrary, development models are best regarded as *a coherent set of practices*, some of which are required to balance the potential performance trade-offs arising from the use (or absence) of others.

9. Conclusions

Overall, our data suggests that anecdotal evidence emerging about the process and practice strengths developing in India are well founded. They also show some continued strengths in Japan. But, as is common in this type of research, due to the extreme variations in performance from project to project, it is hard to draw any definite conclusions. It is important to remember, as well, that no Indian or Japanese company has yet to make any real global mark in widely-recognized software innovation, which has long been the province of U.S. and a few European software firms. Code productivity taken in isolation may not be a good proxy for business performance, and is probably less valuable than a defect measure for judging the performance of a development organization. Unfortunately, comparable financial performance data is almost impossible to come by in these surveys, given the wide range of unique circumstances that each project addresses (e.g., custom developed software for internal use versus packaged software for retail sale). Nonetheless, above all, our data shows that Indian organizations are doing an admirable job of combining conventional best practices, such as specification and review, with more flexible techniques that should enable them to respond more effectively to customer demands. If such a trend is replicated across the broader population, it suggests the Indian software industry is likely to experience continued growth and success in future.

10. References

[1] McConnell, S., *Rapid Development*, Microsoft Press, Redmond, WA, 1996.

[2] Gilb, T., *Principles of Software Engineering Management*, Addison-Wesley, New York, NY, 1988.

[3] Cusumano, M., and C. F. Kemerer, "A Quantitative Analysis of US and Japanese Practice and Performance in Software Development", *Management Science*, vol. 36, no. 11, November 1990, pp. 1384-1406.

[4] Blackburn, J.D.; Scudder, G.D.; Van Wassenhove, L.N. "Improving speed and productivity of software development: a global survey of software developers." *IEEE Transactions on Software Engineering*. December 1996: 875–885.

[5] Cusumano, M. *Japan's Software Factories*, Oxford University Press, New York, 1991.

[6] Matsumoto, Y., "A Software Factory: An Overall Approach to Software Production," in P. Freeman, ed., *Tutorial: Software Reusability*, IEEE Computer Society Press, Washington D.C., 1987.

[7] Zelkowitz, M., et al., "Software Engineering Practices in the U.S. and Japan," *Computer*, June 1984: 57-66.

[8] Iansiti, M., and A. MacCormack, "Developing Products on Internet Time," *Harvard Business Review*, September-October 1997.

[9] Cusumano, M. and R. Selby. "How Microsoft Builds Software" Communications of the ACM, June 1997, vol. 40, no. 6: 53-62.

[10] MacCormack, A. "Product-Development Processes that Work: How Internet Companies Build Software" *MIT Sloan Management Review*, vol. 42 no. 2, 2001, pp. 75-84.

[11] Cheung, P. "Practices for Fast and Flexible Software Development," unpublished master's thesis, MIT Department of Electrical Engineering and Computer Science, June 2002.

[12] Sharma Upadhyayula, "Rapid and Flexible Product Development: An Analysis of Software Projects at Hewlett-Packard and Agilent," unpublished master's thesis in System Design and Management, MIT, June 2001.

[13] MacCormack, A., C. Kemerer, M. Cusumano, and W. Crandall, "Software

Development Practices: Exploring the Trade-offs Between Productivity and Quality," forthcoming, *IEEE Software*.

[14] Cusumano, M. and R.W. Selby. *Microsoft Secrets*, Simon & Schuster, New York, NY, 1998.

[15] Cusumano, M. and D. Yoffie. "Software Development on Internet Time *IEEE Computer*, October 1999, 2-11.

[16] Cusumano, M. *The Software Business*. Free Press/Simon & Schuster, New York, forthcoming 2004.

[17] Kemerer, C.F., An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, 1987. vol. 30 n. 5, 416-429.

[18] Banker, R.D., S.M. Datar, and C.F. Kemerer, A Model to Evaluate Variables Impacting Productivity on Software Maintenance Projects. *Management Science*, 1991. vol. 37, n. 1,. 1-18.

[19] Kemerer, C., MacCormack, A., Cusumano, M. and W. Crandall, "Practice and Performance in Software Development: A Global Study", working paper, 2003.