Journal of Cost Analysis and Parametrics, 7:120–128, 2014
Copyright © ICEAA
ISSN: 1941-658X print / 2160-4746 online
DOI: 10.1080/1941658X.2014.922905

www.iceaaonline.com

Engineering Systems: Best-in-Class/Worst-in-Class

DONALD M. BECKETT

QSM, McLean, Virginia

Measurement's goal is to help assess performance—to determine which methods are productive or counterproductive. Metrics are tools used to identify and implement practices that lower costs, reduce time to market, and improve product quality. But process improvement is not accomplished through measurement or metrics alone. Rather, one must use the data to make conscious decisions that change the way business is done. In fact, one of best ways to make those decisions is by studying the characteristics of best- and worst-in-class software projects. Referencing Quantitative Software Management's database of 10,000+ completed software projects, this article evaluates the common factors that define the most and least successful engineering projects—drawn from the database's System Software, Scientific, Telecom, and Command and Control application domains. Presenting a thorough analysis of project staffing, effort, duration, cost, and quality data, this article gives project managers a solid, scientific framework for evaluating potential projects and identifying winning strategies.

Introduction

Insanity is often humorously defined as what happens when we continue doing things the same way but expect different results. The goal of measurement is to help business leaders and analysts assess performance: to see which methods work and which are counterproductive. Metrics are tools used to identify and implement practices that lower cost, reduce time to market, and improve product quality. But process improvement does not occur automatically as a byproduct of measurement. Rather, it is the product of conscious decisions to change the way we do business.

One way to identify best practices is to study the characteristics of best and worst-in-class software projects. In the 2006 QSM Software Almanac (QSM, 2006), a study was conducted using 564 information technology (IT) projects from the QSM database of 10,000+ completed software projects. This article analyzes engineering projects to see if their best and worst projects tell a similar story.

Engineering class software projects are drawn from the System Software, Scientific, Telecom, and Command and Control application domains. A separate review of these domains concluded that their characteristics were similar enough to treat them as a group for research purposes. Best-in-class and worst-in-class are relative categories: best and worst performers position themselves relative to other projects in the sample being studied. The worst performers in the sample of completed software projects collected metrics and delivered functioning software. Not included in the sample were the "worst of the worst": projects that neither completed nor left forensic evidence to be examined later.

Scatter plot data is available to be shared. Complete records for individual database projects may not be available. Please contact the author for more information or clarification on sharable data.

Address correspondence to Donald M. Beckett, 2000 Corporate Ridge, McLean, VA 22102. E-mail: don_beckett@qsm.com

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/ucap.

Likewise, best-in-class projects are ranked against a select group of projects that had formal measurement programs in place. So in a real sense, they represent "the best of the best" the industry has to offer.

For this analysis, best-in-class projects are defined as projects that perform at least one standard deviation better than the Engineering Systems average for both time to market and effort expended. Worst-in-class projects are just the opposite; their effort and schedule were at least one standard deviation below the industry average. In software development, project effort and time to market often work against each other. When schedules are compressed, projects pay a premium in effort. Projects that outrank their peers in two dimensions—that have best-in-class effort and schedule performance—are relatively rare. Clearly they are doing something right, and the characteristics they share suggest potential best practices. Likewise, worst projects help identify problem areas or counterproductive practices that should be avoided or mitigated.

To evaluate performance, a comparison and contrast was conducted of staffing, effort, project duration, quality, and the amount of concurrent analysis and development activity for best, average, and worst performers. The results do not present a simple checklist of what to do (or not to do) on a project; but they do give the engineering project manager a framework for evaluating projects and identifying risky plans, strategies, and assumptions.

Best-in-Class/Worst-in-Class Process

To identify the best and worst projects, a sample of 276 engineering projects was compared against the QSM engineering trend lines for Size versus Effort and Size versus Duration. FIGURE 1 plots project duration (top graph) and effort (bottom graph) against size in lines of code. The trend lines represent average and + and - 1σ regression fits from the QSM engineering industry trends.

Green circles represent the 13 best-in-class projects, red squares are the 10 worst-in-class, and black dots represent the remaining engineering domain projects that were neither best- nor worst-in-class.

Using the QSM engineering trend lines, values for schedule duration, effort, peak staff, cost, and productivity index (PI) were determined at the average and + and - 1α to determine the differences in schedule and effort between best in class, worst in class, and average projects. The table below summarizes the findings.

TABLE 1 compares typical cost, effort, schedule, staff, and PI values for best-in-class, average, and worst-in-class projects. The schedule and effort numbers for best- and worst-in-class represent the boundaries where those categories begin. Cost was estimated at $10,000 per staff month. The results are striking:

- *Duration*: The duration cutoff for best-in-class is 1.8 times more efficient for time to market than the average and nearly 3.1 times more efficient than the cutoff for worst-in-class projects. These results are very similar to those found in QSM's 2006 study of IT projects (QSM, 2006), in which the best-in-class projects were 1.9 times better than the average and 3.5 times better than the worst-in-class.
- *Effort*: For effort, the difference is even more pronounced: the best-in-class boundary is 2.9 times more efficient than the average and nearly 8.2 times more efficient than the worst-in-class boundary. For IT projects in the 2006 study, the ratios were 2.8 and 8.7 respectively. These boundaries represent a statistical mechanism for defining a range of normal variability for engineering software projects: one that includes roughly 2/3 of the projects. From 1σ above average to 1σ below, schedule

varies by a factor of 3.5 and effort by nearly 9. The best and worst projects exceed these boundaries. Consequently, their differences are even more pronounced.

In software development, tradeoffs between schedule and effort are common. Typically, managers attempt to compress schedules by adding people (effort). Conversely, when cost is the most constraining factor, projects can keep costs down by reducing the

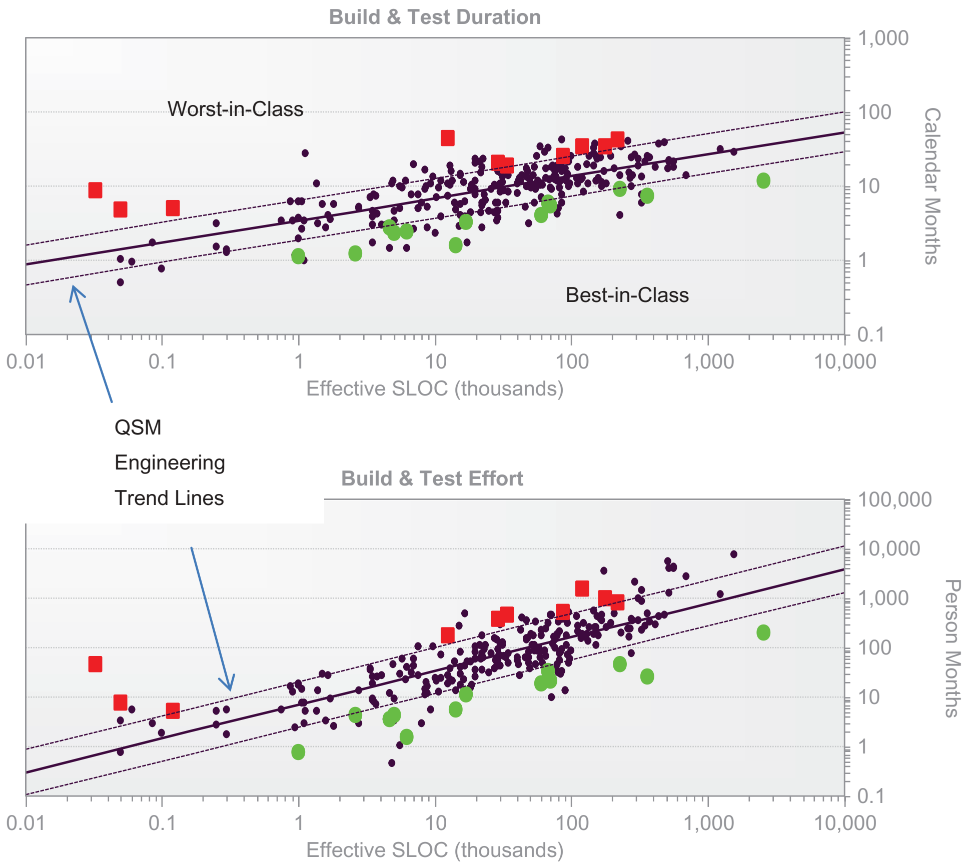


FIGURE 1 Project Duration and Effort vs. Size. Black circles are best-in-class projects, gray squares worst-in-class, and black dots represent the remaining engineering domain projects that were neither best- nor worst-in-class.

TABLE 1 Typical 32,500 K project (averages for schedule, effort, peak staff, cost, and productivity)

Metrics	Best-in-Class	Average	Worst-in-Class
Schedule (Months)	7.5	13.3	23.5
Effort (Months)	34	99	279
Peak staff	6.7	10.7	16.7
Cost ($ Thousands)	340.1	986.0	2,788.3
PI (Productivity)	18.4	13.5	8.6
Project count	13	276	10

team size and stretching out the schedule. But best-in-class and worst-in-class projects are radically different from the normal scenario where time is traded for effort/cost or vice versa. These extreme performers are either very successful or very unsuccessful at minimizing both time to market and cost.

Average project sizes of our best- and worst-in-class data sets differ, too. The best projects are spread over the size spectrum, while the worst projects have three tiny projects, then no projects at all until reaching a project size of about 11,000 lines of code. Notably, about 25% of the engineering samples were small enhancements that fall in that size range, yet only a handful of these small projects were worst performers. This may lend support to the old axiom that keeping projects small and manageable is a software best practice. While keeping projects small does not guarantee success, it does seem to make failure less likely.

Staffing

FIGURE 2 illustrates the staffing profiles of the best- and worst-in-class projects and lends credence to the adage that “too many cooks spoil the meal.” Only one of the worst-in-class projects had a staffing level lower than the engineering systems average and only one of the best-in-class was higher. To put this in perspective, worst-in-class projects used four times as much staff and took three times longer to complete than the best-in-class projects. If there is one takeaway from this analysis, it is to use the smallest team possible to do the work. Large teams do not complete projects significantly earlier. They cost more, are more difficult to manage, and (as will be seen when quality is examined) they create an inferior product.

Worst-in-class projects used far larger teams than the best-in-class but achieved little schedule compression.

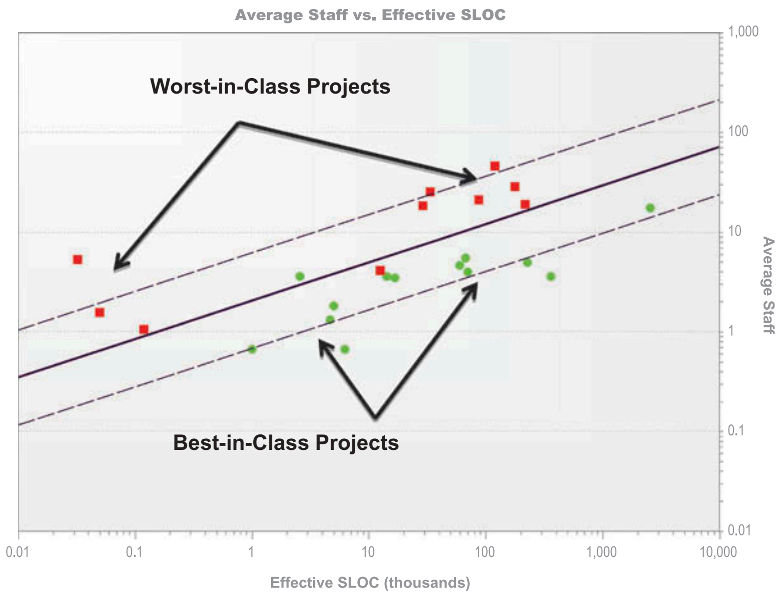


FIGURE 2 Average staff for best- and worst-in-class projects.

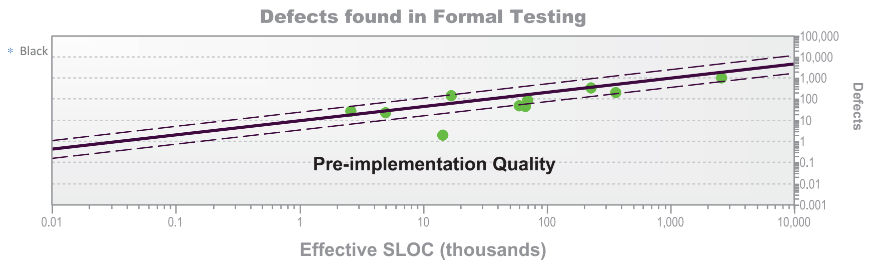
Quality Analysis

For this study, two quality measures were captured: pre-release defects and mean time to defect (MTTD). MTTD measures the average time between discovered errors in the post-implementation product stabilization period. Interpreting defect data is anything but a straightforward task. It can be difficult to tell whether a high number of pre-delivery defects in a project represent poor quality in the product or effective defect identification and removal processes. For this reason, the study uses MTTD too, since defects identified in production are by definition problems. In the 2006 Almanac it was found that in Information Technology, best-in-class projects were far more likely to report defects than the worst-in-class. Engineering projects display the same trend, but the effect is even more dramatic: 85% of the best-in-class projects reported pre-implementation defects while only 10% (one project) of the worst-in-class did so. And that project reported zero defects. Of the remaining engineering projects (ones that are neither best- nor worst-in-class), 62% reported pre-release defects. FIGURE 3 presents the results. Overall, the best-in-class projects reported fewer pre-release defects than their engineering peers.

Worst-in-class projects did not report pre-implementation defects, while the best-in-class were generally better than the engineering systems average. Post-implementation quality of the best-in-class was significantly better than average.

Typically, only a small group of projects report post-implementation defects. This may be attributable to the difficulties projects face in collecting data once a project has been put into production. Frequently, a separate group or organization provides project support. Also, when a project modifies an existing system, it can be difficult to trace defects back to their original source. As a group, the best-in-class projects had significantly higher quality in production than the engineering systems average. The two worst-in-class projects that reported production defects were close to the average for engineering systems.

Defect tracking is not just a nice add-on feature to software development. It provides critical information about the quality of the product being developed and insight into what



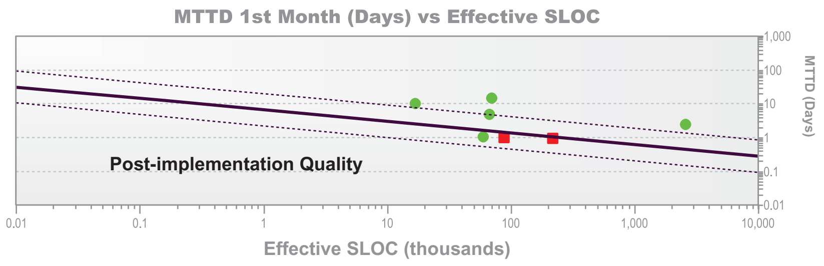


FIGURE 3 Defects during Formal Testing vs. Effective SLOC and MTTD 1st month vs. Effective SLOC.

works well and what requires improvement in the software development process. That the worst-in-class projects had such a poor record for tracking defects points directly to organizational process problems.

Schedule and Effort

The distribution of time and effort over the life cycle of a project affects its overall performance. Projects that spend a higher percentage of their time and effort on the upfront activities of requirements determination, analysis, and design typically expend less effort overall, complete sooner, and have fewer defects. FIGURES 4 and 5 compare the percentages of time and effort the best- and worst-in-class projects expended on various lifecycle activities. Two things stand out: the differences in the initial (feasibility) phase and also in the post-implementation warranty or maintenance phase.

The feasibility phase puts boundaries around a project. It defines the overall project scope and outlines the set requirements to be included or excluded from the project. When done properly it provides an effective framework for software development. The warranty or maintenance phase is the time between project implementation and when it is stable enough to be transitioned to production support.

Surprisingly, best- and worst-in-class projects expended roughly the same relative amounts of schedule and effort on actual development, but allotted different amounts of time and effort on the early and follow-on phases:

- Best-in-class projects spent much more time and effort in feasibility and functional design (the first two phases) than worst-in-class projects: 31.4% versus 22.5% for effort and 43.8% versus 31.5% for schedule.

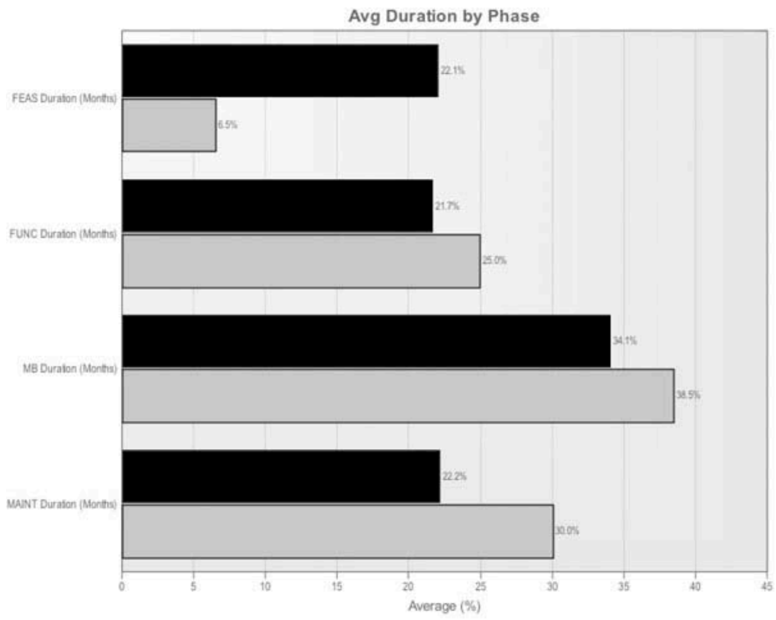


FIGURE 4 Average duration by phase. Black bars are best-in-class projects, gray bars are worst-in-class.

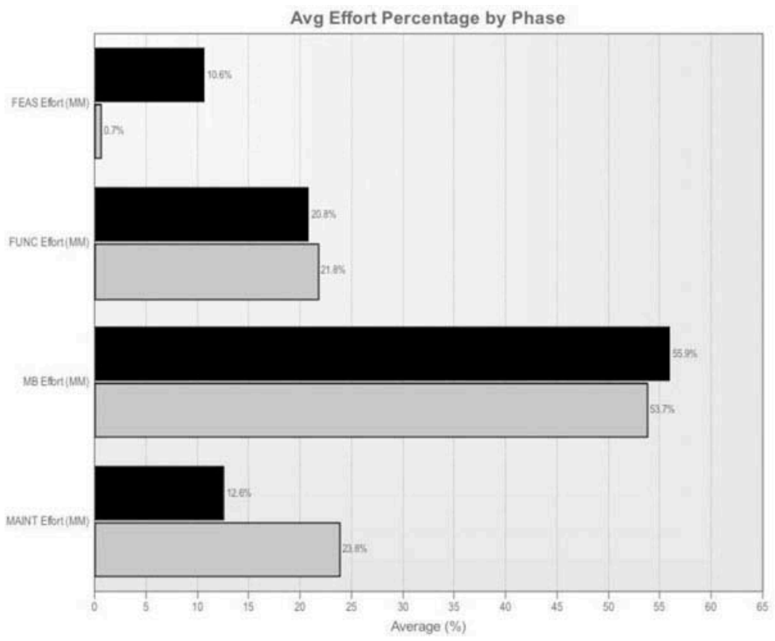


FIGURE 5 Average effort percentage by phase. Black bars are best-in-class projects, gray bars are worst-in-class.

- In the warranty phase, the opposite was true. Worst-in-class projects expended 23.8% of their effort and 30% of their schedule in warranty, while the best-in-class used 12.6% of their effort and 22.2% of their schedule.

In a nutshell, the time and effort spent in up-front activities by the best-in-class projects produced products that required less time and effort to maintain after implementation.

Best-in-class projects spent more time in feasibility and functional design and less in the maintenance (warranty) phase than the worst-in-class, as shown in FIGURE 4.

FIGURE 5 is a good illustration of the "pay up-front or pay later (with interest)" concept. The worst-in-class projects spent a negligible amount of effort in feasibility but paid for it later in maintenance.

Concurrency

In the context of software development, concurrency is a measure of how many activities are performed simultaneously. The overlap between the analysis/design and development was examined to see if there was a difference between best- and worst-in-class projects. As summarized in TABLE 2, the results were noteworthy: worst-in-class projects had significantly more overlap than the best-in-class. What does this mean?

TABLE 2 Overlap between analysis/design and development

	Average % overlap	Median % overlap
Best-in-class	16	0
Worst-in-class	43	40

The hypothesis is that the worst-in-class projects began coding before the requirements were adequately developed and designed. This resulted in increased rework, higher defects and cost, and longer schedules. Please note that this is not a criticism of iterative development or prototyping; rather it is old-fashioned advice to know what you are going to do before starting to do it.

Further Investigation Warranted

The largest single customer of engineering systems analyzed in this study was the U.S. military (28% of the projects in the sample), followed closely by the Aerospace Industry with 23%. Only one best-in-class project was developed for the military, but 80% of the worst-in-class projects were defense projects. What explains this? We can only speculate and offer some observations and suggestions for further research.

Military projects may operate in a more formal environment with more stringent controls than non-military projects, thus increasing overall project time and effort. Team sizes certainly are part of the answer. Military projects were staffed at higher levels than corresponding non-military projects. At the median size of military projects (90 k lines of code) average staff was 2.2 times higher than non-military projects. As FIGURE 2 illustrates, all but one of the worst-in-class projects were staffed at levels exceeding the engineering systems average. An analysis of environmental factors found no evidence that the worst-in-class military projects were more complex undertakings than the other engineering projects.

Conclusions

Every software development manager wants his or her project to succeed. Successful projects satisfy customers and enhance careers. They help win repeat business. What is a successful project? Is it not one that goes according to plan, meets its schedule, completes within budget, and is warmly received by its customer? In this study, characteristics have been identified that are associated with successful projects, and also with unsuccessful ones. Successful engineering systems projects:

- keep scope small and manageable;
- use smaller teams, ones less than average on the QSM trend lines;
- spend sufficient time on up-front requirements determination and analysis before coding;
- allow sufficient time for the project to complete (the term "aggressive schedule" is another term for a project that will cost more and have poorer quality); and
- track defects and analyze them to improve performance.

These are not new concepts, and they do not represent a simple checklist. Doing all of them will not guarantee a best-in-class project, but failure to do them dramatically increases the likelihood of a project being worst-in-class.

The conclusions reached and project behaviors observed in the 2006 study of IT projects are similar to those observed for the engineering domain:

- *Staffing*: Small team size was a characteristic of the best projects in both studies.
- *Defects*: The worst projects did not track defects while the best projects did in both studies. Project quality was above average for both sets of best projects.
- *Time to complete*: Best projects completed in about 1/3 the time of the worst in both studies.

While engineering and information technology projects develop different types of software, they share much in common, and it is fair to say that the characteristics that promote success or failure in them are applicable to software development in general.

Reference

QSM. (2006). *The QSM Software Almanac: 2006 IT Metrics Edition*. McLean, VA: Quantitative Software Management.

About the Author

Donald M. Beckett is a principal consultant at Quantitative Software Management. He has over 30 years of experience in software and has focused on software measurement and estimation since 1995. He previously published a study on Best-in-Class / Worst-in-Class business IT projects in “The QSM Software Almanac, Application Development Series.”