What I Did Last Summer: A Software Development Benchmarking Case Study

James T. Heires, *Rockwell Collins*

This article describes a vendor-supported benchmarking study of an applications development department. The study established a quantitative performance baseline of the organization and compared it to industry trends.

Rockwell Collins, a world leader in the design, production, and support of communications and aviation electronics solutions for commercial and government customers, recently funded a benchmarking study. The study aimed to improve the ability of Rockwell Collins' applications development department to deliver software solutions to its internal customers. The deliverables from this study included current performance, recommendations for improvement, and a database containing the department's historical project performance.

The study concluded that the department as a whole produced applications cost effectively but more slowly than most other organizations in its industry. It further concluded that single-developer Web-based projects performed differently, in several ways, than the larger team-oriented non–Web-based projects. Using root cause analysis, the study identified several factors associated with software development performance.

Based on the results, the consultants—Quantitative Software Management (QSM) Associates in Pittsfield, Massachusetts—recommended enhancing the existing project management processes with requirements management, project management, staff training, and a metrics program. The department planned to implement the recommendations using the practices described in the Software Engineering Institute's Capability Maturity Model for Software (SW-CMM).[1]

The department's smaller projects were characterized by motivated but less experienced developers, short schedules, highly volatile requirements and staffing, high complexity, and few standards and tools. The large projects, on the other hand, were just the opposite. What wasn't understood was the magnitude of the project performance characteristics and what effect they had on development productivity.

The company

Rockwell Collins employs 16,000 people worldwide—the applications development department has about 100 people within a 500-employee IT department (see Figure 1). The applications development department produces traditional and cutting-edge business systems to support every facet of the com-

The study discovered that the majority of the department's software development effort is expended on very large projects.

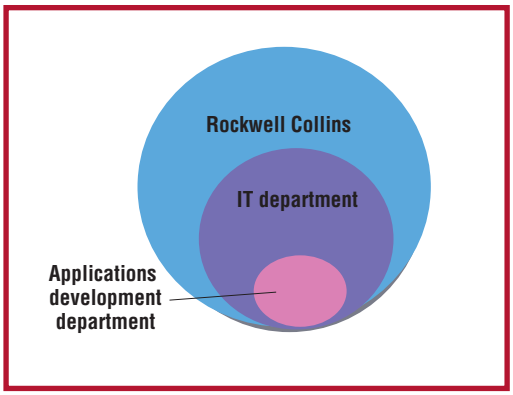

Figure 1. Organizational context.

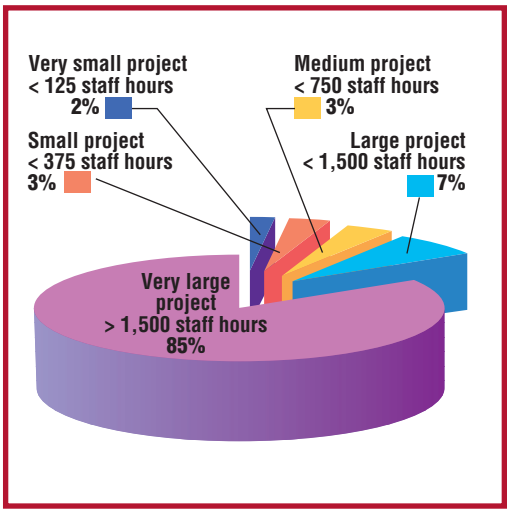

Figure 2. Projects by effort.

pany's products and services, including manufacturing, accounting, marketing, and human resources. The study discovered that the majority (85 percent) of the department's software development effort is expended on very large projects—those requiring more than 1,500 staff hours of effort each (see Figure 2).

The applications development department employs a broad variety of technologies that bring solutions to its customers. Platforms vary greatly from HP and IBM mainframes to client-server to stand-alone PCs to the Web. Of the 28 projects involved in this benchmarking study, 28 different programming languages were employed in various combinations. Some of the more popular languages include Delphi, Visual Page, HTML, Visual Basic, Lotus Notes, Access, Cobol, Pascal, Perl, Power Builder, VB Script, Crystal Reports, Oracle, ABAP, and SQL.

Tools and development environments were equally varied. A declining mainframe development environment was making way for a newer, but less mature, Web infrastructure. The IBM mainframe environment included such rudimentary features as production configuration control, debugging, build control, and program trace, but was managed in a consistent, centralized fashion. Although developers complained about the weak tool set, the consistent process provided a much-needed development standard and discipline. Web development tools, on the other hand, were more modern but varied by programming language. Most development languages came with their own compiler, debugger, database, and testing tools. There were few configuration management, quality assurance, or design tools available across these environments.

The study was undertaken to help quantify the value of the department's software process improvement initiative—choosing the SW-CMM as its improvement model because Rockwell Collins had been successfully using it in other business units. The company planned two benchmarking studies (one early and one late) to quantify the improvements. Although it is usually difficult to connect cause and effect, this dual measurement approach was selected to help quell the expected accusations that the initiative had no impact on the bottom line. The department had recently been assessed at CMM Level 1 and had just put in place a core team of process improvement specialists to help carry out the initiative.

We used QSM Associates to assist with this benchmarking study for several reasons—time being one reason. Because the project schedule was aggressive, there wasn't time to use internal resources. In addition, using a professional benchmarking firm lent more credibility to the study results. QSM Associates had carried out dozens of similar studies over the past 20 years and maintains a large database of historical performance statistics from various industries.

Benchmark methodology overview

Planning for the study began with the consultants and department management establishing boundary conditions, including cost, schedule, deliverables, roles, and responsibilities. These conditions helped provide some management controls.

WBS	Task Name	Duration	Start	Finish
9	**Benchmark Studies – QSM**	326 days	Mon 5/1/00	Mon 8/20/01
9.1	**QSM Benchmark – 1**	**48 days**	**Mon 5/1/00**	**Mon 7/10/00**
9.1.1	Preparatory Data Collection	1 week	Mon 5/1/00	Fri 5/5/0
9.1.2	Kickoff Meeting	1 day	Mon 5/6/00	Mon 5/6/00
9.1.3	Data Collection Interviews	3 days	Tue 5/9/00	Thu 5/11/00
9.1.4	Follow Up	2 weeks	Mon 5/15/00	Fri 5/26/00
9.1.5	Final Report	1 day	Mon 7/10/00	Mon 1/10/00

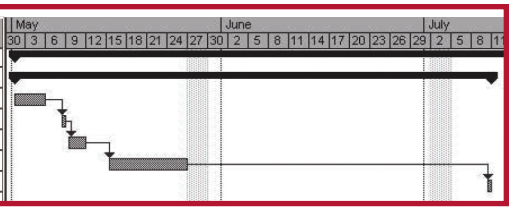

Figure 3. Detailed schedule of benchmarking tasks.

Using a professional benchmarking firm lent more credibility to the study results.

The study's cost consisted of consulting fees and employee time needed to carry out the study. The benchmarking was part of a larger project and was represented by a series of tasks in the project schedule (see Figure 3). The deliverables the consultants provided included a kick-off presentation, a set of findings and recommendations, a final presentation, and a database of our own historical project data. In addition, the consultants demonstrated how to leverage the database to better estimate new projects. They also established roles and responsibilities to minimize the study's duration and management oversight required. The project manager planned the study, selected and negotiated with the consultants, identified target projects and their key personnel, ensured the study performed to plan, and provided logistics support. The consultants provided preparatory materials, manpower to carry out the data collection interviews, data validation and statistical analysis of the data, and kick-off and final presentations. Key project team members from participating projects gathered project data, attended data collection interviews, and supported post-interview data validation activities.[2]

The department generated a list of approximately 75 recently completed projects to be included as part of the benchmarking study, then narrowed that list down to 30 to 50 projects. To perform a reasonable quantitative analysis, statisticians recommend a sample size of at least 30 data points. Trends are easier to determine and outlying data points have a reduced effect on the analysis results.

The consultants knew in advance that some candidate projects would not be on the final list. They disqualified projects for several reasons, including incomplete or unobtainable core metrics, unavailable knowledgeable project team members, and prematurely cancelled projects.

Because the projects with complete data were more likely to be chosen, they were also more likely to exhibit more discipline and therefore higher productivity. However, it was also likely that other organizations represented in the reference database had a similarly biased distribution. Who, after all, would submit mediocre project data to be included in an industry-wide database? Thus, the comparison between the department's performance and that represented by the industry reference database is appropriate.

The consultants provided preparation materials—data collection forms, metrics definitions, and guidelines. Next, the benchmark project manager asked the other PMs and developers to gather preliminary data on their projects. The requested data included the Software Engineering Institute's core measures of size, time, effort, and defects.[3] Although several PMs expressed concern about participating in the study, an early decision to explicitly fund the study helped convince some project teams to participate. Other project teams exhibited stereotypical shyness about having their project performance measured. In my experience, this is quite common in lower-maturity organizations that have less experience with software project measurement or less understanding of its benefits.

The on-site benchmarking study began with a brief kick-off presentation given by the consultants to invigorate management and set the participants' expectations for the remainder of the study. The consultants explained the process as well as the deliverables and their value.

Data collection

Immediately following the kick-off meeting, three solid days of data collection began. A 90-minute time slot was allocated to each project, giving each PM a chance to tell his or her story. The PM and one or two key developers from the project brought their prepared project data to the data collection

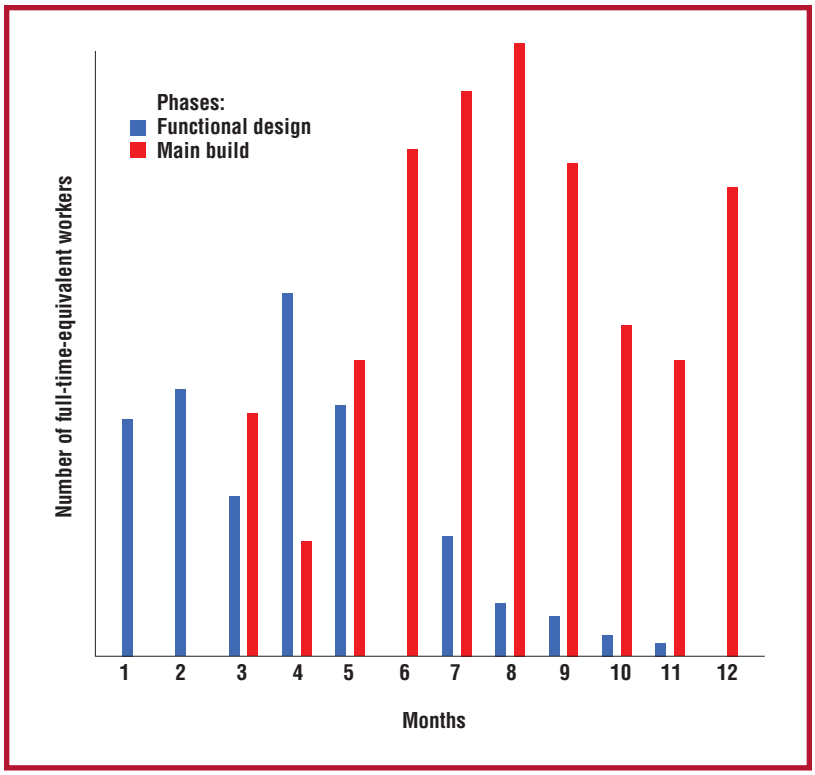

Figure 4. Staffing profile by phase.

interview. A consultant and a process improvement specialist also attended.

For each project, data collection began with SEI core measures. These quantitative measures included the project's size, time, effort, and defects. Additional qualitative and demographic data were also collected—application complexity, team skills, development environment, and technology used.

For a Level 1 organization like us, this data was difficult to gather. Some projects did not establish a separate time-charging mechanism to allow measuring effort separately for each project. Other projects did not know when the project began because of numerous false starts.

To facilitate gathering this critical data, a simple staffing profile was produced early in the interview (see Figure 4). Drawing a monthly (or weekly) timeline showing the number of full-time-equivalent people working on the project helped the team remember what happened. Different colors indicated each phase, illustrating any phase overlap. This simple chart helped us determine phase-specific effort and duration information.

In a perfect world, each PM would have arrived at the interview with a complete set of data to enter into the database. In reality, however, this was a painful and slow process, requiring several passes. More than half of the projects on the initial list had to be dropped from the study because of incomplete or inaccurate data. One issue that plagued data collection on several projects was PM staff turnover. Some projects ratcheted through two or three PMs during their six-month life cycle. The lack of a data collection infrastructure was also a cause of "project amnesia," whereby the entire project legacy left with the departing PM.

The QSM data structure

QSM Incorporated, a sister organization to QSM Associates, has a well-established data structure and associated analysis methodology that is employed at benchmarking engagements. The data structure originated with an empirical study carried out by QSM researcher and chief scientist Larry Putnam.[4] Putnam's study established a software equation that defined a metric called the Productivity Index in terms of size, time, and effort. This macro-level metric is primarily used to empirically determine a project's, or an organization's, relative development efficiency. The QSM data structure includes the SEI's recommended core measures as well as post-delivery defects, planned duration, planned effort, the number of requirements, the languages used, and demographic data (for example, project team members, application type, and tools used).

The consultants needed to carry out some simple transformations in order to use the QSM data structure. Size was the most difficult conversion. The two most popular size measures—source lines of code (SLOC) and function points—can be used directly in any combination in the QSM data structure. Because we did not have trained individuals in function point counting, we relied on physical measures such as modules, programs, and SLOC as our fundamental size measures. Line-based languages (for example, Pascal, ABAP, and HTML) could be counted directly, while other languages were converted to SLOC. Languages with means of expression other than lines of code (for example, Visual Basic, IMG, Crystal Reports, and Lotus Notes) were converted to SLOC. Several QSM-provided gearing factors en-

abled conversion from these native measures to SLOC. Gearing factors are expressed as a range of values. For example, one frequently used language is Crystal Reports. These reports are made up of fields, which, on average, tend to be between 2 and 10 SLOC each, depending on complexity.

Although QSM derived its gearing factors empirically through benchmarking studies such as this one, converting size units reduces the size measure's precision. Using function points was an option for this study, and we used them on a few projects as data validation. However, function points still must be converted to SLOC in order to use QSM's data model. Furthermore, it would have been difficult to use function points in this stage of our maturity because of our lack of expertise in that area. In retrospect, SLOC was the best unit of size for this situation.

Another adjustment required to conform to QSM's data structure involved the development phase definitions. Data is requested by phase of development: feasibility study, functional design, main build (detailed design through delivery), and maintenance. If a project used life-cycle models, such as the rapid-prototyping or evolutionary paradigms, the data had to be reconfigured to match the QSM phases. This was not, however, a difficult task. It consisted of combining effort and schedule information from the project data into the development phases defined by QSM.

Although the SEI and other organizations have tried to foster standard measurement definitions, such definitions must be better defined and more widely used before the industry truly benefits. Core measures need standard definitions and empirical studies to help establish their use in business. One problematic example is the inadequate definition of SLOC. The Software Productivity Consortium has one of the only documented definitions.[5] Unfortunately, the SPC definition is quite dated and does not adequately define common relationships such as the number of changed lines of code. With standards in place, tools would evolve to support the standard.

All project data found its way into QSM's data collection and analysis tool, Slim-Metrics.[6] This database was instrumental in ensuring that project data was collected in a consistent fashion, and it greatly facilitated the analysis portion of the benchmarking study.

The most important step in the process is validating the collected data for correctness and consistency. Without valid data, the analysis is of little value. The consultants carried out this step, but the project team members supported it. As the data was collected, and when analysis began, issues about the data's validity arose. As these issues were clarified, confidence in the data's validity improved. The validation step was completed only after the data was scrutinized and cross-checked. The consultants then had a good, quantitative understanding of what happened on the project.

Data analysis

The benchmarking study gathered and validated data from 28 projects. The data was stratified in multiple dimensions to uncover strengths and weaknesses. Root cause analysis was the basic method used, but various statistical, trend, and demographic analyses were also part of this activity.

Root cause analysis relies heavily on the experience of the analyst and attempts to find causes that explain the organization's behavior. Positive and negative causes were sought to deliver a balanced recommendation. Various analytical tools were used to find the root causes, including stratification, scatter plots, Pareto diagrams, and histograms.

The analysis becomes the department's legacy, so it should be representative. It explains the quality, efficiency, size, and cost of the organization's software development projects. The analysis is the basis of a deliverable—consisting of an annotated presentation and a database of project data—and shared with the organization. The database is later used to support further analysis, project estimates, and tracking activities.

The findings

The response to the benchmarking study was positive and immediate. Developers and senior management alike resonated with the study's implications. Everyone involved felt the study accurately characterized the software department's development capability.

Although the study didn't reveal any surprises, it did quantify our behavior. The applications development department expends

The benchmarking study gathered and validated data from 28 projects. The data was stratified in multiple dimensions to uncover strengths and weaknesses.

The most immediate benefit received was the establishment of a performance baseline for the company from which it could measure improvements.

much less effort and staffing than the industry average on their projects, compared to like-sized projects in QSM's IT database. Consequently, the project duration is generally longer than average. The Productivity Index of the 28 projects followed the industry trend line but had much more volatility. We decided to stratify the data to identify the performance differences of various project types.

Stratification proved to be the most revealing part of the analysis. When completed, two significant factors emerged: team size and Web technology. Compared to non–Web-based, team development projects, Web-based single-developer projects exhibited the following positive and negative differences.

The positives were

- 10 percent more efficiency (as measured by the Productivity Index),
- 64 percent lower cost,
- 45 percent shorter duration,
- 47 percent higher field reliability, and
- 77 percent smaller team size.

The negatives were

- 7.5 percentage points more scope growth,
- 19 percent more staff turnover,
- 65 percent less experience on the application being developed,
- 53 percent more logic and 28 percent greater logic complexity,
- 11 percent more customer issues, and
- 24 percent lower development tool capability.

Some of the benefits of benchmarking your own organization are enhanced software estimates, better project tracking, and proof of the value of improvement initiatives.

The vast majority of software estimation models were developed with the use of empirical data collected from hundreds or thousands of actual projects. However, all estimation models should be calibrated to an organization's own software development behavior before being used to forecast a new project. The difference between a calibrated and an uncalibrated estimation tool is immense. A properly calibrated estimation model generates more accurate results than an uncalibrated one.[7] That being said, calibration is best carried out only after a thorough quantitative study. Using a calibrated estimation model serves as a sanity check against the estimate. The historical data helps answer the all-important question, "compared to what?" Forty-eight percent of the projects in the study used various Web-based technologies. If an estimate of a new Web site were needed, it might make sense to use an estimation model calibrated with these projects and to compare the results with the historical project data.

Benchmarking data can also enhance tracking and oversight of in-flight projects. If, for example, your project is entering the testing phase but the original deadline has already elapsed and the customer is asking for an estimated delivery date, benchmarking data can help. Using a defect discovery rate for your project and a calibrated forecasting model, you can make this task much easier. Perhaps more importantly, you can avoid the embarrassment of delivering a defect-laden product too early.

Evidence of the benefits of improvement can often be difficult to demonstrate. Many times, however, historical data and a few statistical techniques can prove that value has been delivered as a result of an improvement initiative.[8] Although this initiative is not yet complete, the existence of this benchmarking study will make the post-improvement analysis easier.

The department benefited from this study in many ways. Because the company was just starting its process improvement journey, the most immediate benefit received was the establishment of a performance baseline for the company from which it could measure improvements.

Another benefit was the ability to compare our performance with industry trends. This comparison let us determine how competetive we are in our industry.

In addition, the final analysis listed recommendations that included processes to improve the practices of requirements management, project planning, staff training, and metrics collection. These recommendations were combined with the goals of the SW-CMM initiative to constitute an action plan for improvement. The action plan was

a critical step for the department, which had never before based an improvement plan on actual project performance data.

Next summer, the company plans to repeat the study and compare the two data sets. This comparison will clearly show how much improvement the company realized over the duration.

About the Author

James T. Heires is the principal project manager at Rockwell Collins. His professional experiences include design of electronic flight instrumentation systems, engine indicator and crew alerting systems, flight management systems, and consumer electronics. He is working to improve the state of the practice of project management through parametric cost estimation and quantitative tracking techniques. Contact him at jtheires@rockwellcollins.com.

References

1. M.C. Paulk, C.V. Weber, and B. Curtis, *The Capability Maturity Model: Guidelines for Improving the Software Process*, SEI Series in Software Engineering, Addison-Wesley, Reading, Mass., 1995.
2. *IT Organization, Benchmark Thyself,* Cutter Consortium, Arlington, Mass., 2000, pp. 29–31 and 67–71.
3. A. Carleton et al., *Software Measurement for DOD Systems: Recommendations for Initial Core Measures*, tech. report CMU/SEI-92-TR-019, ADA 258 305, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1992.
4. L.H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Trans. Software Eng.*, vol. SE-4, no. 4, 1978, pp. 345–361.
5. R. Cruickshank and J. Gaffney, *Code Counting Rules and Category Definitions/Relationships*, Version 02.00.04, Software Productivity Consortium, Herndon, Va., Apr. 1991.
6. J.T. Heires, "Measuring QSM's Repository and Analysis Tool," *Application Development Trends Magazine*, vol. 5, no. 6, 1998; www.ADTmag.com/Pub/jun98/pr601-2.htm (current 15 Aug. 2001).
7. B. Boehm et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, Upper Saddle River, N.J., 2000, pp. 150–151.
8. J.T. Heires, "The High Technology Historian: Historical Data Analysis," *IT Metrics Strategies*, vol. VI, no. 8, Aug. 2000.