



## The Business of Software

### An Updated Software Almanac

*Research into what makes software projects succeed.*

**S**OFTWARE PROJECTS CAN be so complicated and so different from each other that predicting whether they will succeed or fail can be as difficult as forecasting the weather or picking winning stocks. Will the project entirely fulfill its goals? Will it deliver some value at a higher cost or later than desired? Or will it just crash and burn leaving the exhausted survivors to lick their wounds, bury the dead bodies, and shred the evidence?

Courageous efforts are being made to collect and codify the data that is available, to try to spot what trends are occurring in the industry, and to provide some useful guidelines for managing the business of software. The recently published *QSM Software Almanac*, dubbed the “2014 Research Edition,” is an example of this.

#### QSM Software Almanac

Quantitative Software Management (QSM) published the IT Metrics Edition of its Software Almanac in 2006.<sup>1</sup> This highly detailed analysis of thousands of project data points was interesting reading and reached a few quite jaw-dropping conclusions. The 2014 version confirms much of the 2006 analysis and provides further insights. It begins by suggesting how software projects and organizations should be measured and what we can infer from these measurements.



#### Five Core Metrics

In software projects and organizations, the list of things-that-could-be-measured is daunting. The intricacy of projects, the differences in the types of systems being built, in project life cycles, in toolsets, in expertise and development environments all add to the variables operating in our business. But some simple core metrics, available from all projects and enterprises, can be as important as the measurement of temperature and blood pressure are to medical diagnosis. They may not diagnose all

ailments, but they can give provide insights into the health of projects and organizations.

These metrics<sup>2</sup> are:

**Schedule**—the elapsed calendar time from a project start until completion of its mandate;

**Effort or cost**—which is usually driven by the number of staff employed during the project duration;

**Functionality delivered or “system size”**—this is the *value component* of the project, what is obtained for the cost and schedule expended;

**Quality or defect level**—the “anti-

size” or that portion of the system that does not work properly on delivery; and

**“Productivity”**—the rate at which the resources of time, effort, and staff are turned into delivered functionality (minus defects).

The *QSM Software Almanac* uses these relatively simple metrics, backed up with more sophisticated measurements, to make some important points about modern systems development:

**Projects seldom measure performance.** Most projects in QSM’s database measure and report schedule, effort, and size, but only one-third of the projects actually use this data to assess their performance. This is a missed opportunity for improvement.

**Sacrifice schedule first.** Projects are willing to overrun schedules more than budget and are more willing to overrun budget than deliver less functionality to customers.

**Big teams are bad.** In this study we see repeatedly that larger teams cost a lot more, deliver lower-quality systems, but seldom save time. In one quoted example, cutting project staff from 100 people to 52 people saved \$12 million but resulted in only a modest increase in schedule. Worst-in-class projects, as measured by poor performance and low quality, are strongly correlated with large team size and best-in-class projects are strongly correlated with small team sizes. In addition, large-staffed projects tend to try to fix any problems they encounter by adding even more staff whereas small-staffed projects usually try other, more effective, approaches. QSM’s data also showed a wide range of staffing for similarly sized systems; this may be driven by the common practice of throwing people at projects in trouble. Smaller-staffed projects showed between three and eight times greater productivity than larger staffed projects delivering equivalent value(!).

**Quality is getting better over time.** This is especially true for engineering systems, which are getting larger while IT systems are getting smaller and taking longer. This was shown for best-in-class projects; worst-in-class projects seldom collect quality data, but probably have worse quality.

**Reuse does not usually work.** “Minor enhancement” projects with high lev-

els of expected reuse typically cost twice that of equivalent new development.

**Agile advantage tops out around 30K line-of-code equivalents.** The data shows that, at their present stage

of evolution, Agile methods are best suited to smaller projects. This finding seems to support a general industry perception of Agile. But QSM also found that large Agile projects have

Figure 1. Typical engineering project duration.

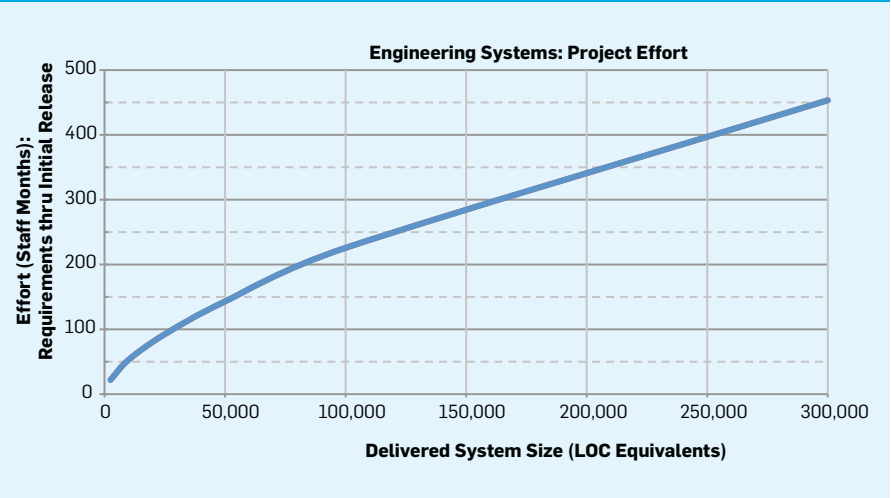


Figure 2. Typical engineering project effort.

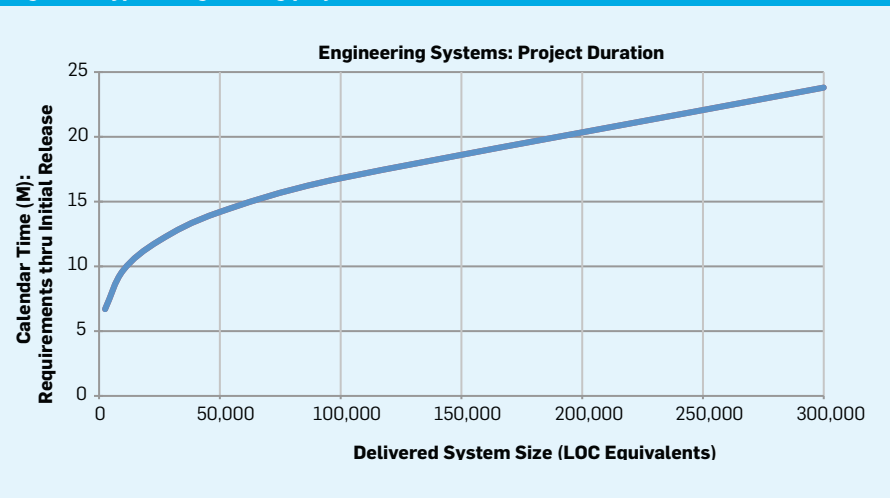
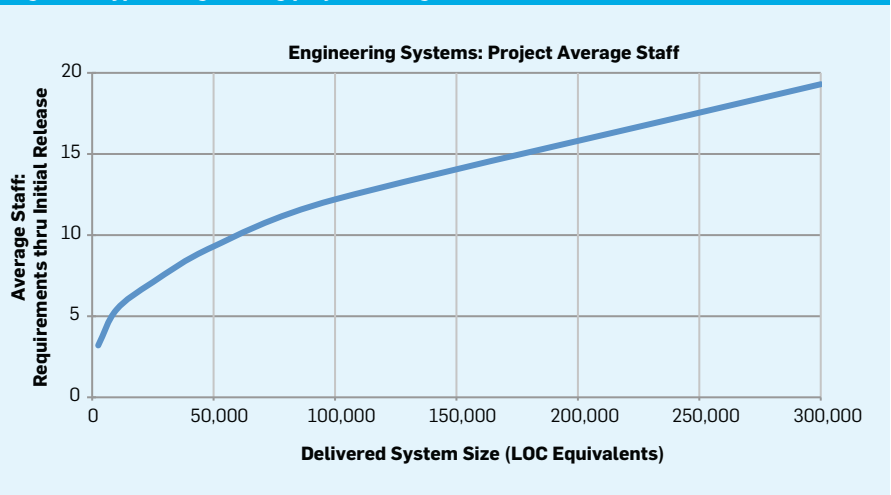


Figure 3. Typical engineering project average staff.





Association for  
Computing Machinery

## ACM Conference Proceedings Now Available via Print-on-Demand!

*Did you know that you can now order many popular ACM conference proceedings via print-on-demand?*

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

**For available titles and ordering info, visit:**  
[librarians.acm.org/pod](http://librarians.acm.org/pod)



## In software projects and organizations, the list of things-that-could-be-measured is daunting.

much more variable results, so it is likely the benefits of larger Agile projects depend a lot on how they are implemented. Also, larger Agile projects that experience problems are better off adding time than dropping functionality or adding staff.

**Projects grow.** Average functionality-to-be-delivered increases 15% over the life of a project, typically creating schedule overruns of 8% and cost overruns of 16%.

### Some Industry Trends Over Time

The Almanac shows:

- ▶ The median project duration contraction that occurred during the 1990s and 2000s has leveled off to 10–11 months.
- ▶ Project effort is declining along with the delivered size of systems, but delivered size is varying more as time goes by.
- ▶ Median project team size is relatively stable.
- ▶ There is a strong trend toward more financial systems development in IT.
- ▶ Mixed-language development is the norm, but Java has become the most common development language.

### Measurement Ideas

Some of the articles in the Almanac describe useful approaches for the measurement of projects. One article describes the use of Shewart control charts (modified for Rayleigh mathematics) to calibrate defect discovery trends and use them for predictive purposes. Another covers four models for data mining that can be used for process improvement. A third article describes an empirical study to see what project and environmental factors most correlate with schedule performance on projects. For business applications

these factors are overall technical system complexity and team communication complexity—an important finding for distributed or offshore teams.

### Performance Benchmark Tables


The Almanac's final set of tables can be used to determine if your project fits close to typical measured norms for business, engineering, or real-time systems development. Given a reasonable assessment of delivered system size, these tables would allow a project manager to determine typical values for project duration, effort/cost, and average staff in a minute or two, and avoid serious overcommitments in time, money, staff, or delivered functions (see figures 1–3 for typical engineering system ranges).

### Lessons Learned

This study indicates some simple lessons:

- ▶ Collect the basic and practical core metrics on projects and use them to figure out what you can do, what you are doing, and how well you are doing it.
- ▶ Commit projects at levels around measured industry benchmarks in duration, effort, and staff.
- ▶ Avoid using large teams—they will probably be ineffective.
- ▶ Do not add people to a project that hits snags—it will likely make things worse; instead consider giving the project more time.
- ▶ Do not assume reuse will save you anything in time or effort.
- ▶ Manage larger Agile projects carefully so the benefits are not nullified by size and complexity.

### It's Free

The *QSM Software Almanac 2014 Research Edition* contains a great many more insights and detailed analyses of the copious amount of data collected. It is free for download from QSM at: <http://bit.ly/1y5MahV>. 

### References

1. Armour, P.G. Software, hard data. *Commun. ACM* 49, 9 (Sept. 2006), 15–17.
2. Putnam, L.H. and Myers, W. *Five Core Metrics*. Dorset House Publishing Co., 2003.

**Phillip G. Armour** ([armour@corvusintl.com](mailto:armour@corvusintl.com)) is a vice president at Applied Pathways LLC, in Schaumburg, IL, and a senior consultant at Corvus International Inc., in Deer Park, IL.

Copyright held by author.