# Project Management Tools and Software Failures and Successes

Capers Jones
*Software Productivity Research, Inc.*

*The construction of large software systems is one of the most hazardous activities of the business world. The failure or cancellation rate of large software systems is over 20 percent. Of the large systems that are completed, about two thirds experience schedule delays and cost overruns that may approach 100 percent. About the same number are plagued by low reliability and quality problems in the first year of deployment. Yet, some large systems are finished early, meet their budgets, and have few if any quality problems. How do successful projects differ from projects that fail? Better project management and better quality control are the most important differences between success and failure in the software world.*

Software development is a troubling technology. Software is highly labor-intensive, and as a result, large software projects are among the most expensive undertakings of the 20th century. Large software systems cost far more to build and take much longer to construct than the office buildings occupied by the companies that have commissioned the software. Extremely large software systems in the 100,000 function point size range can cost more than building a domed football stadium, a 50-story skyscraper, or a 70,000-ton cruise ship.

Consider what the phrase "large systems" means in the context of six different size plateaus separated by an order of magnitude for each plateau. Size is expressed in terms of function points, a widely used synthetic metric based on five external attibutes of software applications: inputs, outputs, inquiries, logical files, and interfaces. The average number of C statements found within the typical function point is provided as a point of reference.

## **One Function Point** (125 C Statements)

There are few software applications of this size except small enhancements to larger applications or minor personal applications. The schedules for such small programs are usually only from a day to perhaps a week.



## **10 Function Points** (1,250 C Statements)

This is the typical size of end-user applications and also a tremendously frequent size plateau for enhancements to existing software. Development schedules are usually less than one month.

## **100 Function Points** (12,500 C Statements)

This size is heavily populated with enhancements to existing applications. It is also the practical upper limit of end-user applications. There are few stand-alone applications of this size in 1998, but 10 years ago there were a number of DOS applications in this size range, such as early BASIC interpreters. However, there are many features of larger applications that approximate this size. Development schedules are usually less than six months. Individual programmers can handle applications of this size, although technical writers and other specialists may be involved, too.

## **1,000 Function Points** (125,000 C Statements)

This size range exceeds the capabilities of end-user development. This is a fairly common entry-level size range for many commercial and internal Windows software applications. It is also a common size range for in-house client-server applications. Schedules for software projects of this size are usually longer than 12 months. In this size range, the volume of specifications and user documentation becomes a significant contributor to software costs.

Quality control also is a major requirement at this size range. Applications of this size range require development teams of up to 10 staff members, since individual programmers cannot usually handle the volume of code and other deliverables. Specialties such as quality assurance, technical writing, and database administration are often represented on the development team. With team development, issues of system segmentation and interfaces among components become troublesome.

## **10,000 Function Points** (1,250,000 C Statements)

Applications of this size are usually termed "systems" because they are far too large for individual programs. This size range is often troubled by cost and schedule overruns and by outright cancellations. Development teams of 100 or so are common, so communication and interface problems are endemic.

Software schedules in this size plateau run from three to more than five years, although the initial planning for applications of this size range tends to naively assume schedules of 18 months or less. The volume of paperwork in terms of plans, specifications, and user manuals is so large that production of documents is often more expensive than the source code. Because defect levels rise with application size, formal quality control including pre-test inspections are

necessary for successful completion. Configuration control and change management also are mandatory for this size plateau.

## 100,000 Function Points (12,500,000 C Statements)

Applications that approach 100,000 function points in size are among the most troubling constructs of the 20th century. This is roughly the size range of Microsoft's Windows 95 product and also IBM's MVS operating system. This is also the size range of major military systems.

Software development schedules for systems of this size are usually from five to more than eight years, although the initial development plans tend to assume 36 months or less. Development teams number in the hundreds, often in multiple locations that may even be in different countries. Communication problems are rampant. Paperwork and defect removal operations will absorb the bulk of development costs. Formal configuration control and change management are mandatory and expensive for this size plateau.

Using these six size ranges, Table 1 shows the approximate frequency of various kinds of outcomes, ranging from finishing early to total cancellation. Table 1 is taken from *Patterns of Software Systems Failure and Success* (International Thomson Computer Press, 1996).

As can easily be seen from Table 1, small software projects are successful in the majority of instances, but the risks and hazards of cancellation or major delays rise quite rapidly as the overall application size goes up. Indeed, the development of large applications in excess of 10,000 function points is one of the most hazardous and risky business undertakings of the modern world.

## Software Successes and Disasters Within Six Subindustries

There are six major subindustries within the software community that tend to follow somewhat different practices and even use different tools and programming languages. In terms of their ability to successfully build large software applications, these six subindustries in order of rank are

- Systems software.
- Outsource vendors.
- Commercial software.
- Military software.
- Management information software.
- End-user software.

It is interesting to consider why there are variances among these industries in the ability to complete large software projects. These six categories are the most common types of software development projects in North America, South America, Europe, Africa, India, the Middle East, and the Pacific Rim.

### Systems Software

This category refers to applications that control physical devices such as operating systems, navigation and flight control, telecommunication systems, process control systems, automotive fuel injection, medical instruments, and the like. The systems software community is concerned with software that operates large and complex physical devices. If quality is not excellent, then the devices may fail during use; therefore, the systems software community has learned the hard way that careful quality control is on the critical path. The systems software community, overall, has the best track record for building large software applications. This community also has the best quality control and the best suites of quality control tools.

### Outsource Vendors

These are companies such as Andersen Consulting, Computer Sciences Corporation, Electronic Data Systems, IBM's Integrated Systems Solutions, and a number of others. These companies build software under contract for their clients. As a class, the outsource vendors are often better equipped and better trained than the clients they serve. This is not always true, but if it were not true, fairly often, the outsource business would fail. The outsource community often has highly sophisticated project management and quality control tool suites available, significant amounts of reusable material, and highly trained personnel.

### Military Software

These are applications constrained to follow various military standards such as the older DOD-STD-2167A standard or the newer MIL-STD-498. Military applications that control weapons systems tend to resemble civilian systems software projects in terms of the emphasis on careful planning and quality control.

The military and defense community is not in reality bad at building large systems, but there is a major problem in this domain. Military standards are so complex and baroque that the productivity of defense applications is lower than any other software subindustry. The reason for low productivity has nothing to do with coding or technical work. Military standards trigger such enormous volumes of paperwork that there are roughly 400 English words created for every Ada statement on military software projects. The volume of paperwork on military software projects is almost three times that of comparable civilian projects of the same size.

Table 1. *Software project outcomes by size of project.*

| | PROBABILITY OF SELECTED OUTCOMES | | | | |
|---|---|---|---|---|---|
| | Early | On Time | Delayed | Canceled | Sum |
| 1 FP | 14.68% | 83.16% | 1.92% | 0.25% | 100.00% |
| 10 FP | 11.08% | 81.25% | 5.67% | 2.00% | 100.00% |
| 100 FP | 6.06% | 74.77% | 11.83% | 7.33% | 100.00% |
| 1,000 FP | 1.24% | 60.76% | 17.67% | 20.33% | 100.00% |
| 10,000 FP | 0.14% | 28.03% | 23.83% | 48.00% | 100.00% |
| 100,000 FP | 0.00% | 13.67% | 21.33% | 65.00% | 100.00% |
| Average | 5.53% | 56.94% | 13.71% | 23.82% | 100.00% |

| | Systems Software | Military Software | MIS Software | Outsource Software | Comm. Software | End-User Software | Average |
|---|---|---|---|---|---|---|---|
| 1 FP | 99.00% | 98.00% | 98.00% | 98.00% | 99.00% | 95.00% | 97.83% |
| 10 FP | 96.00% | 93.00% | 95.00% | 97.00% | 98.00% | 75.00% | 92.33% |
| 100 FP | 88.00% | 84.00% | 86.00% | 88.00% | 89.00% | 50.00% | 80.83% |
| 1,000 FP | 75.00% | 65.00% | 68.00% | 74.00% | 75.00% | 5.00% | 60.33% |
| 10,000 FP | 54.00% | 38.00% | 30.00% | 47.00% | 35.00% | 0.00% | 40.80% |
| 100,000 FP | 28.00% | 15.00% | 5.00% | 24.00% | 10.00% | 0.00% | 18.40% |
| Average | 73.33% | 65.50% | 63.67% | 71.33% | 67.67% | 37.50% | 65.09% |

Table 2. *Probability of on-time software delivery in six subindustries.*

### Commercial Software

This refers to the high-volume shrink-wrapped packages by companies such as Borland, Computer Associates, and Microsoft. Until recently, this subindustry did not build many large applications, so they have had some catching up to do. As the size of personal computer software packages approaches the size of mainframe software packages, the commercial vendors have had to increase their project management tools and methods, strengthen quality control, and in general, imitate the successful pattern of the systems software domain.

### Management Information Systems (MIS)

This refers to the internal applications companies build for their own use: accounting systems, payroll systems, insurance claims handling, banking and financial systems, etc. The MIS community does not have a particularly good track record when it comes to large systems. Often, the MIS community lags in quality control and testing technologies compared to the other communities. However, project management tools for MIS companies are now increasing in number and capability.

### End-User Software

This refers to applications built privately by people for their own use, which in the context of this article means applications used for business or professional purposes, not games or home applications. Although tools such as Visual Basic, Realizer, spreadsheets, and SAS have expanded the capabilities of the end-user community, there is still a low upper limit to the sizes of applications that end users can construct. About 100 function points is the practical upper limit and 1,000 function points is the current maximum size of end-user applications.

## Probabilities of On-Time Software Delivery, Cancellations, or Delays

The first summary topic of interest is the probability that software projects will be finished on time, using the initial schedule estimate derived during requirements as the basis of the comparison. Table 2 shows the on-time rates but needs some explanation first. There is an anomaly in the data because there are no end-user applications larger than 1,000 function points; therefore, the 0 percent values in the end-user column are excluded from the average values. Also, some projects finish early, but these are included in the on-time percentages. The probability of an early finish for 10,000 function points or larger is approximately 0 percent.

As can be seen, small software projects are comparatively well controlled within all six subindustries. As the overall size ranges grow larger, delays and cancellations become much more common and also more severe. On the whole, the systems software community and the outsource community have the best results with systems in the 10,000 to 100,000 function point range; the military domain comes in third place.

### Probability of Termination

The next topic of interest is the probability that a project will be terminated prior to completion. This is among the most severe risks we face in software—only termination with accompanying litigation is more disastrous. Table 3 shows the probabilities of software project terminations for the various subindustries.

To illustrate our failures on an intuitive level, consider the following analogy: If building construction had the same ratio of cancellations as software, more than half the office buildings in the world larger than 30 stories tall would be abandoned before completion. The average height of buildings in New York City would be only three stories, and there would be no skyscrapers.

None of the six domains have fully mastered the ability to construct large software systems without a significant risk of termination or cancellation. However, the systems software community and the outsource community have the best track record for large systems, with the military software community coming in third. The information systems community fails repeatedly for large systems. The commercial software world is not particularly good at the large system plateau—though it is getting better—and end users cannot do large systems.

Table 3. *Probability of software project termination in six subindustries.*

| | Systems Software | Military Software | MIS Software | Outsource Software | Comm. Software | End-User Software | Average |
|---|---|---|---|---|---|---|---|
| 1 FP | 0.10% | 0.10% | 0.10% | 0.10% | 0.10% | 1.00% | 0.25% |
| 10 FP | 1.00% | 2.00% | 1.00% | 1.00% | 2.00% | 5.00% | 2.00% |
| 100 FP | 5.00% | 7.00% | 6.00% | 6.00% | 5.00% | 15.00% | 7.33% |
| 1,000 FP | 12.00% | 15.00% | 17.00% | 14.00% | 9.00% | 65.00% | 22.00% |
| 10,000 FP | 25.00% | 33.00% | 45.00% | 40.00% | 45.00% | 100.00% | 48.00% |
| 100,000 FP | 40.00% | 55.00% | 80.00% | 45.00% | 70.00% | 100.00% | 65.00% |
| Average | 13.85% | 18.68% | 24.85% | 17.68% | 21.85% | 47.67% | 24.10% |

| | Systems Software | Military Software | MIS Software | Outsource Software | Comm. Software | End-User Software | Average |
|---|---|---|---|---|---|---|---|
| 1 FP | 0.90% | 1.90% | 1.90% | 1.90% | 0.90% | 4.00% | 1.92% |
| 10 FP | 3.00% | 5.00% | 4.00% | 2.00% | 0.00% | 20.00% | 5.67% |
| 100 FP | 7.00% | 9.00% | 8.00% | 6.00% | 6.00% | 35.00% | 11.83% |
| 1,000 FP | 13.00% | 20.00% | 15.00% | 12.00% | 16.00% | 30.00% | 17.67% |
| 10,000 FP | 21.00% | 29.00% | 25.00% | 13.00% | 20.00% | 0.00% | 11.20% |
| 100,000 FP | 32.00% | 30.00% | 15.00% | 31.00% | 20.00% | 0.00% | 16.60% |
| Average | 12.82% | 15.82% | 11.48% | 10.98% | 10.48% | 14.83% | 10.81% |

Table 4. *Probability of schedule slip by more than 25 percent in six subindustries.*

### Probability of Schedule Overrun

The next topic of interest is the probability that a software project will eventually be finished but will run later than anticipated by a significant amount (a 5 percent slip is noticeable, more than 10 percent is painfully costly, and a 50 percent slip is a catastrophe). The initial estimate developed during requirements is the starting point. Table 4 shows slippage probabilities for the six subindustries.

Here, too, the low end of the software size spectrum is generally trouble-free and under full control. As the size range gets larger, delays and cancellations become much more common. A contributing factor to both delays and cancellations also grows with size: The probability of "creeping user requirements." The average growth of unplanned, unanticipated requirements is about 1 percent to 2 percent per month during the design and coding phases of typical software projects, although the upper range of requirements creep can exceed 10 percent in a single month.

Any of the six domains can build small software projects with a good probability of success. At the upper end, no domain is fully capable. However, the systems software world, the large outsource contractors, and the military domains are the most experienced with large applications and therefore have somewhat better probabilities of succeeding.

## Project Management Tools Used on Successful Software Projects

One of the newer uses of the function point metric is to evaluate the completeness of various kinds of software tool suites. This approach can clearly reveal some of the critical differences between successful software projects, average projects, and total failures.

It is obvious to consultants who spend much time with large systems and large corporations that manual methods are not adequate for cost estimation, schedule planning, or quality prediction. The best-in-class organizations may have more than 10 times the quality tool capacities and more than 30 times the project management tool capacities than the organizations that fail with software.

Interestingly, there may be little if any difference in the capacities of software engineering tool suites. Both successful and unsuccessful companies tend to have in the range of 30,000 to perhaps 50,000 function points of software engineering and development tools. The difference between companies that succeed and those that do not is that the former employ effective project management tool suites whereas the latter generally do not. Table 5 identifies the typical patterns of project management tools noted on leading, average, and lagging software projects.

As shown, the lagging projects tend to be essentially manual for most project management functions. The leading projects deploy a notable quantity of quality control and project management automation. Leading projects tend to use more than 16 times the project management tool capacities of lagging projects in terms of function points. In terms of numbers of project management tools deployed, there is about a 6-to-1 ratio between the leading and lagging projects.

The presence of a suite of project management tools is not, by itself, the main differentiating factor between successful and unsuccessful software projects. The primary reason for the differences noted between lagging and leading projects is that the project managers who use a full suite of management tools are usually better trained and have a firmer grasp of the intricacies of software development than the managers who lack adequate management tools.

Bringing a large software project to a successful conclusion is an extremely difficult task filled with complexity. The managers who can deal with this complexity recognize that some of the cost and resource scheduling calculations exceed the ability of manual methods. Managers on failing projects, on the other hand, tend to have a naive belief that project planning and estimating are simple enough to be done using rough rules of thumb and manual methods.

## Summary and Conclusions

Software is intangible, but the schedules and cost estimates for software can be highly tangible. Software projects are still subject to the basic laws of manufacturing, and software needs to be placed on a

Table 5. *Numbers and size ranges of project management tools (size data expressed in terms of function point metrics).*

| Project Management | Lagging | Average | Leading |
|---|---|---|---|
| Project planning | 1,000 | 1,250 | 3,000 |
| Project cost estimating | | | 3,000 |
| Statistical analysis | | | 3,000 |
| Methodology management | | 750 | 3,000 |
| Year 2000 analysis | | | 2,000 |
| Quality estimation | | | 2,000 |
| Assessment support | | 500 | 2,000 |
| Project measurement | | | 1,750 |
| Portfolio analysis | | | 1,500 |
| Risk analysis | | | 1,500 |
| Resource tracking | 300 | 750 | 1,500 |
| Value analysis | | 350 | 1,250 |
| Cost variance reporting | | 500 | 1,000 |
| Personnel support | 500 | 500 | 750 |
| Milestone tracking | | 250 | 750 |
| Budget support | | 250 | 750 |
| Function point analysis | | 250 | 750 |
| Backfiring: LOC to FP | | | 750 |
| Function point subtotal | 1,800 | 5,350 | 30,250 |
| Number of tools | 3 | 10 | 18 |

firm engineering basis by the end of the 20th century.

Project managers are the primary key to software project success and failures. To a large degree, the sophistication or lack of sophistication of the project management tool suite will determine whether software projects will succeed, experience major cost and schedule overruns, or fail. ◆

## About the Author

Capers Jones is an international consultant on software management topics and chairman of Software Productivity Research, Inc. (SPR) in Burlington, Mass. He began his software career as a programmer in the Office of the Surgeon General, Washington, D.C. Prior to becoming chairman of SPR, he worked at the Crane Company, IBM, and was assistant director of programming technology at ITT Corporation's Programming Technology Center in Stratford, Conn.

Software Productivity Research, Inc.
1 New England Executive Park
Burlington, MA 01803-5005
Voice: 781-273-0140
Fax: 781-273-5176
E-mail: capers@spr.com

## Suggested Readings

1. Brown, Norm, ed., *The Program Manager's Guide to Software Acquisition Best Practices*, Version 1.0, U.S. Department of Defense, Washington, D.C., July 1995.
2. Charette, Robert N., *Software Engineering Risk Analysis and Management*, McGraw-Hill, New York, 1989.
3. Charette, Robert N., *Application Strategies for Risk Analysis*, McGraw-Hill, New York, 1990.
4. DeMarco, Tom, *Controlling Software Projects*, Yourdon Press, New York, 1982.
5. DeMarco, Tom, *Why Does Software Cost So Much?*, Dorset House, New York, 1995.
6. Department of the Air Force, *Guidelines for Successful Acquisition and Management of Software-Intensive Systems*, Vols. 1 and 2, Software Technology Support Center, Hill Air Force Base, Utah, 1994.
7. Dreger, Brian, *Function Point Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1989.
8. Grady, Robert B., *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, Englewood Cliffs, N.J., 1992.
9. Grady, Robert B. and Deborah L. Caswell, *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
10. IFPUG Counting Practices Manual, Release 4, International Function Point Users Group, Westerville, Ohio, April 1995.
11. Jones, Capers, *Applied Software Measurement*, McGraw-Hill, New York, 2d ed., 1996.
12. Jones, Capers, *Critical Problems in Software Measurement*, Information Systems Management Group, 1993.
13. Jones, Capers, *Software Productivity and Quality Today – The Worldwide Perspective*, Information Systems Management Group, 1993.
14. Jones, Capers, *Assessment and Control of Software Risks*, Prentice-Hall, 1994.
15. Jones, Capers, *New Directions in Software Management*, Information Systems Management Group.
16. Jones, Capers, *Patterns of Software System Failure and Success*, International Thomson Computer Press, Boston, Mass., December 1995.
17. Jones, Capers, *Software Quality – Analysis and Guidelines for Success*, International Thomson Computer Press, Boston, Mass., 1997.
18. Jones, Capers, *The Economics of Object-Oriented Software*, Software Productivity Research, Burlington, Mass., April 1997.
19. Jones, Capers, *The Year 2000 Software Problem – Quantifying the Costs and Assessing the Consequences*, Addison-Wesley, Reading, Mass., 1998.
20. Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, Reading, Mass.
21. Howard, Alan, ed., *Software Metrics and Project Management Tools*, Applied Computer Research, Phoenix, Ariz., 1997.
22. Mertes, Karen R., *Calibration of the CHECKPOINT Model to the Space and Missile Systems Center Software Database*, thesis, AFIT/GCA/LAS/96S-11, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, September 1996.
23. Multiple authors, *Rethinking the Software Process*, (CD-ROM), Miller-Freeman, Lawrence, Kan., 1996. (This is a new CD-ROM book collection jointly produced by the book publisher, Prentice-Hall and the journal publisher, Miller-Freeman. This CD-ROM disk contains the full text and illustrations of five Prentice-Hall books: *Assessment and Control of Software Risks* by Capers Jones, *Controlling Software Projects* by Tom DeMarco, *Function Point Analysis* by Brian Dreger, *Measures for Excellence* by Larry Putnam and Ware Myers, and *Object-Oriented Software Metrics* by Mark Lorenz and Jeff Kidd.)
24. Putnam, Lawrence H., *Measures for Excellence – Reliable Software On Time, Within Budget*, Yourdon Press - Prentice-Hall, Englewood Cliffs, N.J., 1992.
25. Putnam, Lawrence H. and Ware Myers, *Industrial Strength Software – Effective Management Using Measurement*, IEEE Press, Los Alamitos, Calif., 1997.
26. Rubin, Howard, *Software Benchmark Studies for 1997*, Howard Rubin Associates, Pound Ridge, N.Y., 1997.
27. Stukes, Sherry, Jason Deshoretz, Henry Apgar, and Ilona Macias, *Air Force Cost Analysis Agency Software Estimating Model Analysis*, TR-9545/008-2, Contract F04701-95-D-0003, Task 008, Management Consulting & Research, Inc., Thousand Oaks, Calif., Sept. 30, 1996.
28. Symons, Charles R., *Software Sizing and Estimating – Mk II FPA (Function Point Analysis)*, John Wiley & Sons, Chichester, England, 1991.
29. Thayer, Richard H., ed., *Software Engineering and Project Management*, IEEE Press, Los Alamitos, Calif., 1988.
30. Umbaugh, Robert E., ed., *Handbook of IS Management*, 4th ed., Auerbach Publications, Boston, Mass., 1995.
31. Zells, Lois, *Managing Software Projects – Selecting and Using PC-Based Project Management Systems*, QED Information Sciences, Wellesley, Mass.