# Let the Numbers Do the Talking

Donald J. Reifer
*Reifer Consultants, Inc.*

*This article provides software cost and productivity benchmarks for 12 application domains that readers can use to determine how well their organization is doing relative to industry averages, and whether their software estimates are reasonable. In addition to addressing common questions raised relative to the benchmarks, this article summarizes the relative improvement that firms within the applicable industries are experiencing, which range from 8 percent to 12 percent annually.*

For the past 30 years, I have been playing the numbers game. I have been developing benchmarks to confirm that the estimates I was developing were reasonable and achievable. To win the game, I have had to present the numbers in such a way that people I deal with would use them, not abuse them.

Everyone who works in the software business seems to be looking for numbers of one kind or another. I get at least one call a day asking me questions like, "What's the productivity that you've seen across the United States for software within the telecommunications domain?" or "What's the average cost/source line of code for a military system?"

My initial reaction is to try to avoid answering these questions. Why? Because I am afraid that whatever I say will be misquoted. Worse, I am afraid that the numbers I supply will be misused. When pressed for an answer, I express the number as a range. I qualify my answer by saying: " The average cost/line for a military system used for command and control varies between one and three hours per source line where an hour is expressed as directly chargeable labor, and a source line of code is defined using the Software Engineering Institute's counting framework as a logical line. Furthermore, the effort associated with this estimate is scoped to include requirements analysis, architectural design, development, and software integration and test tasks. It does not include system or beta testing, but does include support for requirements analysis."

**"I have used the numbers to win the schedule and budget battles, to acquire investment dollars, ... and most importantly to win management's trust."**

Sounds like a bunch of double-talk doesn't it? Well, it isn't. Nine out of 10 times, no matter what I say, the number is still misquoted or used out of context.

Needless to say, I am getting tired of being misquoted. In response, I have decided to write this article to put some of the more important numbers that I use in the public domain. Why? Well, for two reasons. First, I believe the community could use these numbers as benchmarks. They could serve as industry application domain norms against which organizations could compare their results to determine how well (or not so well) they are doing. Second, I want to get the community thinking about discussing and sharing "like" numbers. That is where I believe the real benefits lie.

Think about it. When push comes to shove, what really matters to management are the numbers. Again, let me give you an example. Suppose you are trying to get your bosses to invest in a new software-engineering environment. They will want justification. Typically, their decision whether or not to fund your proposal will revolve around whether money is available and the answers to the following questions: Will this investment save us money? If so, how much? What are the tax implications? Can we depreciate the equipment and software? If so, can we use either declining balance or straight-line depreciation schedules? What is the projected payback period and return on investment? Is this return higher than those who propose other alternative uses for the money?

Unfortunately, most of the engineers I have worked with during the years haven't the foggiest notion how to answer these questions. The net result is that their proposals are more often rejected than accepted because they fail to prepare a winning business case. However, the same engineers could improve their prospects of winning by using numbers to justify their proposals in terms of productivity improvement, cost reduction/avoidance, quality improvement, and/or time-to-market reduction strategies [1]. They could make the numbers sing to management.

## Making Sense of the Data

My firm has been collecting cost, productivity, and quality data for more than two decades. These data are provided by organizations in exchange for benchmarks that they use for the following major purposes:

- Determine how the organization is doing relative to industry averages within an application domain (automation, command and control, telecommunications, etc.).
- Check the reasonableness of competitive bids.

**Acronym List for Tables**

| | |
|---|---|
| IOC | Initial Operational Capability |
| IPT | Integrated Product Team |
| IRR | Internal Requirements Review |
| KSLOC | Thousands Source Lines of Code |
| LCA | Life Cycle Architecture (review) |
| LCO | Life Cycle Objectives (review) |
| MBASE | Model-Based Software Engineering |
| PDR | Preliminary Design Review |
| PRR | Product Readiness Review |
| SAR | Software Acceptance Review |
| SDR | System Design Review |
| SETD | Systems Engineering and Technical Direction |
| SLOC | Source Lines of Code |
| SM | Staff Month |
| SRR | Software Requirements Review |
| STR | Software Test Review |
| UTC | Unit Test Review |

- Assess organizational shortfalls relative to the competition.

Occasionally, we publish snapshots of these databases [2]. When we do, we get lots of questions and feedback because interest is high, and the need for the numbers is great. To ensure the validity of the data submitted to us, we first screen and then normalize it using definitions that we have developed for that purpose. Once the numbers are entered into our databases, the data are checked for outliers and tested for homogeneity. Standard statistical regression techniques are then used to analyze the data and test it for sampling and other types of statistical errors.

The database has evolved to include data from approximately 1,500 projects. To maintain currency, none of the data retained in the database is more than 10 years old. We continuously refresh the database to purge older projects from it because they tend to bias the results of our analyses. We also remove outliers from the database because they tend to bias the results.

Over the years, my firm's databases have been the source of data for major software cost and productivity studies. For example, the U.S. Air Force relied heavily on the databases to develop a business case for its move to using Ada [3]. In addition, other organizations have used information from these databases to prepare knowledge bases and to calibrate their software cost models. Independent of the points Dr. Randall Jensen raised in his recent CROSSTALK article [4] on the topic of model calibration, we still believe and have the data to demonstrate that a calibrated cost model outperforms one that has not been calibrated. In addition, cost models seem much more accurate than activity-based costing done using Delphi approaches for large projects [5]. While critics of software cost models seem to abound [6], none of those throwing stones at the models have proposed alternative means that have higher prospects of accuracy.

## First Look at Software Productivity

Software productivity refers to the ability of an organization to generate outputs (software systems, documentation, etc.) using the inputs or resources it has at its disposal (people, money, equipment, tools, etc.). Using this definition, an organization can increase its productivity by either focusing on the input or output part of the equation.

| Application Domain | Number Projects | Size Range (KSLOC) | Avg. Productivity (SLOC/SM) | Range (SLOC/SM) | Example Applications |
|---|---|---|---|---|---|
| Automation | 55 | 25 to 650 | 245 | 120 to 440 | Factory automation |
| Command & Control | 43 | 35 to 4,500 | 225 | 95 to 330 | Command centers |
| Data Processing | 36 | 20 to 780 | 330 | 165 to 500 | Business systems |
| Environment/Tools | 75 | 15 to 1,200 | 260 | 143 to 610 | CASE tools, compilers, etc. |
| Military-Airborne | 38 | 20 to 1,350 | 105 | 65 to 250 | Embedded sensors |
| Military-Ground | 52 | 25 to 2,125 | 195 | 80 to 300 | Combat information center |
| Military-Missile | 14 | 22 to 125 | 85 | 52 to 165 | Guidance, navigation and control systems |
| Military-Spaceborne | 18 | 15 to 465 | 90 | 45 to 175 | Attitude control systems |
| Scientific | 33 | 28 to 790 | 195 | 130 to 360 | Seismic processing systems |
| Telecommunications | 48 | 15 to 1,800 | 250 | 175 to 440 | Digital switches and PABX |
| Trainers/Simulations | 24 | 200 to 900 | 224 | 143 to 780 | Virtual reality simulators |
| Web | 64 | 10 to 270 | 275 | 190 to 975 | Client/server sites |
|  | 500 | 10 to 4,500 |  | 45 to 975 |  |

Table 1: *Software Productivity (SLOC/SM) by Application Domains*

### Table 1 Notes

- The 500 projects taken from our database of more than 1,500 projects were completed within the last seven years by any of 38 organizations. (Each organizations' identity is anonymous due to the confidentiality of the data.)
- The scope of all projects starts from software requirements analysis and finishes with completion of software testing.
  - For military systems, the scope extends from software requirements review until handoff to the system test bench.
  - For Web systems, the scope extends from product conception to customer sell-off.
- This includes all directly chargeable engineering and management labor involved.
  - It includes programming, task management, and normal support personnel.
  - It does not include quality assurance, system or operational test, and beta test personnel.
- The average number of hours/staff month assumed was 152.
- SLOC is defined by Florac and Carleton [7] to be a logical source line of code using the conventions published by the Software Engineering Institute in 1993.
- Function point sizes were converted to SLOC using backfiring factors published by the International Function Point Users Group (IFPUG) in 2000, as available on their Web site.
- Different life-cycle models and methodologies are assumed. For example, Web projects typically followed a Rapid Application Development process and used lightweight methods, while military projects used more classical processes and methods.
- Different languages were used. For example, Web projects employed Java and Visual C while military projects used Ada and C/C++.

An input-based strategy would accentuate increasing workforce productivity via efficiencies gained by inserting better methods, tools, processes, facilities, and equipment and collaboration facilities. In contrast, an output-based strategy would place emphasis on reducing the amount of output required by using component technology, product lines, and architecture-centric reuse to eliminate a part of the work involved in developing the product.

Within many industries, productivity is commonly expressed as either source lines of code (SLOC)/staff month (SM) or function points (FP)/SM. Of course, the measures SLOC, FP, and SM must be carefully scoped and defined for these metrics to convey consistent meaning. In addition, there are many factors or cost drivers that cause each of these metrics to vary widely. These must be normalized when defining the terms. Because the Reifer Consultants, Inc. databases are primarily SLOC-based, we use this metric as the basis for our analysis. For those interested, we backfire the supplied FP data in our database using language conversion factors supplied by the International Function Point Users Group (IFPUG) to convert from FP to SLOC (e.g., one FP is expressed as so many lines of C++ or Java).

Table 1 summarizes the results of our analyses for 12 application domains for which we have collected data that we feel are of interest to the CROSSTALK community. The numbers in Table 1 were derived by taking a 500-project subset of our database and performing statistical analysis using various statistical tools. In addition, there are no foreign projects in the database to distort conclusions. Because our clients often challenge our numbers, we pay a great deal of attention to the tendencies and purity of our database. We cannot afford not to do this.

| Application Domain | Ada83 | Ada95 | C/C++ | 3GL | Norm | Notes |
|---|---|---|---|---|---|---|
| Automation | * | * | 30 | 45 | 30 | Most implement ladder nets |
| Command & Control | 70 | * | 50 | 100 | 75 | |
| Data Processing | 25 | 25 | 20 | 30 | 30 | Most have moved to using Java and visual languages |
| Environment/Tools | 25 | * | 25 | 30 | 25 | |
| Military-Airborne | 150 | 125 | 125 | 225 | 175 | |
| Military-Ground | 75 | 75 | 50 | 90 | 75 | |
| Military-Missile | 150 | * | * | 250 | 200 | |
| Military-Spaceborne | 150 | * | 150 | 200 | 175 | |
| Scientific | 75 | * | 65 | 85 | 75 | |
| Telecommunications | 50 | 35 | 40 | 80 | 55 | Most use C/C++ and Unix |
| Trainers/Simulations | 50 | * | 35 | 75 | 50 | |
| Web | * | * | * | * | * | Most use Java and visual languages |

* Not enough data

Table 2: *Software Cost ($/SLOC) by Language by Application Domain*

## Table 2 Notes

- Dollars used to determine cost are assumed to be constant year 2000 dollars.
- The cost assumed per staff month of $12,000 assumes a labor mix and includes direct labor costs plus allowable overhead. This mix assumed that the average staff experience across the team in the domain was three years based on the average staff experience with the application and languages, methods, and tools employed to engineer it.
- Many languages were used in these applications; most projects used more than one language. For example, C/C++ was used heavily in the telecommunication domain, while Java was used extensively for Web applications.
- In addition, to keep costs down many organizations are trying to exploit commercial off-the-shelf packages and large legacy software systems written in languages like COBOL, FORTRAN, Jovial, and PL/1.

## What About Software Cost Numbers?

While cost and productivity of software are related, they are separate considerations when dealing with numbers. To illustrate this point, I have seen several organizations increase their productivity and costs at the same time. In these cases, the organizations were very productive at generating software to the wrong requirements, or building and releasing products with lots of latent defects. That is why we focus attention on each set of numbers separately.

When analyzing our database, we find that software cost tends to be related to both the labor rates and language level (i.e., refers to the methods, tools, and language technology used by the project). To develop numbers of interest, we use $12,000 as the standard cost for a SM of effort exclusive of profit and general and administrative charges, as applicable. Table 2 shows the dollar cost per SLOC by application domains that we have developed as benchmarks.

Table 3: *Waterfall Paradigm Effort and Schedule Allocations*

| Phase (end points) | Effort % | Duration % |
|---|---|---|
| Plans and Requirements (SDR to SRR) | 7 (2 to 15) | 16 to 24 (2 to 30) |
| Product Design (SRR to PDR) | 17 | 24 to 28 |
| Software Development (PDR to UTC) | 52 to 64 | 40 to 56 |
| Software Integration and Test (UTC to STR) | 19 to 31 | 20 to 32 |
| Transition (STR to SAR) | 12 (0 to 30) | 12.5 (0 to 20) |
| Total | 107 to 131 | 116 to 155 |

Note: Percentage allocations for effort and duration are shown as ranges.

Table 4: *Rational Unified Process (RUP) Effort and Schedule Allocations*

| Phase (end points) | Effort % | Duration % |
|---|---|---|
| Inception (IRR to LCO) | 5 | 10 |
| Elaboration (LCO to LCA) | 20 | 30 |
| Construction (LCA to IOC) | 65 | 50 |
| Transition (IOC to PRR) | 10 | 10 |
| Total | 100 | 100 |

Note: Percentage allocations of effort and duration are normalized to 100 percent per the reference [10].

Table 5: *MBASE Paradigm Effort and Schedule Allocations*

| Phase (end points) | Effort % | Duration % |
|---|---|---|
| Inception (IRR to LCO) | 6 (2 to 15) | 12.5 (2 to 30) |
| Elaboration (LCO to LCA) | 24 (20 to 28) | 37.5 (33 to 42) |
| Construction (LCA to IOC) | 76 (72 to 80) | 62.5 (58 to 67) |
| Transition (IOC to PRR) | 12 (0 to 20) | 12.5 (0 to 20) |
| Total | 118 | 125 |

Note: Percentage allocations of effort and duration are shown as ranges and are not normalized per the reference [11].

## How Is the Effort Distributed?

As most of us have learned, distribution of effort and schedule is a function of the life-cycle paradigm (i.e., a modeling method for the software process) selected for the project. In addition, as Fred Brooks [8] so nicely has explained, in many cases, effort and schedule cannot be interchanged. For the following three popular life-cycle paradigms, the allocations of effort and schedule are shown in the Tables 3, 4, and 5:

- Waterfall [9].
- Rational Unified Process (RUP) [10].
- Model-Based Software Engineering (MBASE) [11].

Formats in the tables vary because the numbers are based on slightly modified public references. In some cases, the allocations of effort (and schedule) are shown as ranges. In other cases, they are normalized so that they sum to 100 percent. In yet other cases, the sums are not normalized and therefore equal more than 100 percent.

Tables 3 and 5 clearly show the effort and duration to perform a particular activity relative to what is required for what they consider normal software development (e.g., excludes preparing software requirements and system test tasks). Table 4 reflects the effort and duration to perform tasks that are part of the RUP. Do not infer that MBASE takes 18 percent longer than RUP using these Tables. That is not the case because different life cycles embody different activities that make comparisons between them difficult and misleading. In any case, the results are what is important, not its format. If you are interested in more detailed comparisons between life cycles, see the appendices in [11], which provide the most complete coverage I have seen.

These allocation tables are very revealing. They tell us that software people do much more work than what is considered by most to be software development. This workload typically starts with analyzing the software requirements (i.e., these are typically developed by some other group like marketing or systems engineering) and ends with software integration and test. From Table 3, we can see that software people also take part in developing the requirements, which takes on average 7 percent additional effort and 16 percent to 24 percent more time, and they support system testing, which takes 12 percent more effort and 12.5 percent more time. This makes software cost estimates low by at least 19 percent and schedule estimates short by about 28.5 percent. It is no wonder that software people feel shorted when the budgets are allocated.

## What About the Other Support Costs?

Let us look at the typical costs software organizations spend supporting other engineering organizations. For example, they might participate as members of some integrated product team tasked with developing requirements for the project. Because this effort takes time and effort, funds to perform the work involved need to be estimated, budgeted, and controlled.

As another example, software people will be called upon to assist during systems test and evaluation. They might be called upon to either conduct software beta testing or fix software, hardware, or systems problems during bench, system, and/or operational testing. That is where software shines. It is used to make the system work.

What is important is that these activities consume effort and take talent away from the mainline software development tasks. That means all of these activities need to be thoroughly planned, estimated, budgeted, scheduled, staffed, directed, controlled, and managed by the software manager as the project unfolds.

There is a lot of controversy over how much effort is needed to perform different types of support tasks. Based on experience [12], Table 6 shows examples of how much effort is expended in providing needed support expressed as an average and a range.

Because Independent Verification and Validation and System Engineering Technical Direction contractors increase the development contractor's workload in military contracts, they require additional effort to support such relationships. This is especially true when military organizations use a federally funded research and development contractor to perform system integration and task-direction type tasks.

| Support Cost Category | Effort (% of software development costs) | Notes |
|---|---|---|
| Requirements synthesis and IPT participation | 10 (6 to 18) | Participation in systems definition and specification activities. |
| Systems integration and test | 30 (0 to 100) | Supporting problem resolution activities as the system is integrated and tested. |
| Repackaging documentation per customer requirements | 10 (0 to 20) | Repackaging documentation to meet some customer preference instead of that dictated by the approved organizational process. |
| Formal configuration management (CM) | 5 (4 to 6) | Support to system level CM activities (version control & support for the Software Change Control Board are already included in the software estimate). |
| Independent software quality assurance (SQA) | 5 (4 to 6) | SQA when done by a separate organization. |
| Independent Verification & Validation or SETD support contractor | 6 (4 to 10) | Development organization support to an independent contractor hired to perform technical oversight and provide direction. |
| Total | 66 (18 to 160) | |

Table 6: *Typical Software Support Costs*

### Table 6 Notes

- Percentage allocations of effort are shown as both an average and a range in parentheses.
- Version control of software deliverables is included in our other numbers as is normal quality assurance activities. The additional effort illustrated by Table 6 refers to formal activities like project-level change-control boards that organizations must support.
- If there are subcontracts involving software, additional effort must be added to Table 6 to provide for software support in this area.
- If large amounts of commercial off-the-shelf software will be used, additional effort must be added to Table 6 to support the evaluation, glue code development, integration, and other life-cycle tasks that the software organization will have to support.
- Percentage expressed uses software development costs as its basis. For example, the chart states that the cost on average quoted for a software job should be 166 percent of the base (the base plus 66 percent) to cover the additional costs associated with the support identified in Table 6 when all the categories listed are applicable.
- These support costs can vary greatly based on the manner in which the firm is organized and the work allocated. For example, some firms combine their CM and SQA support in a single Product Assurance organization. In such cases, the average cost for both is 7 percent (6 percent to 8 percent) because they take advantage of economies of scale.
- System integration and test support do not include operational test and evaluation for military projects and beta testing for commercial projects. These can require even more support be applied than that identified. For example, we have seen aircraft projects that have gone through extensive flight testing burn thousands of hours of software support during two- and three-year time periods. As another example, we have seen commercial projects also burn thousands of hours supporting beta testing at remote user sites.
- Many firms have adopted defined processes at the organizational level and are rated as a Level 3 using the Software Engineering Institute's Software Capability Maturity Model® (SW-CMM®) [13]. When this is true, these firms generate documentation as a normal part of their engineering processes. If the customer wishes to reformat this documentation, there will be a repackaging cost because this is not the normal way these firms conduct their business.

## Have We Made Progress?

It is also interesting to look at the trends associated with productivity and cost. Based on the data we analyzed, the nominal improvement firms experience across industries is from 8 percent to 12 percent a year. Those who invest more, typically gain more. For example, jumping a single CMM level can reduce software development costs from 4 percent to 11 percent based on a recent Ph.D. dissertation done by Brad Clark at the University of California [14].

Because I plan to write a separate article on this topic in the near future, I will not clutter this article with additional trend data. However, it is important. For example, a 10 percent improvement in productivity can be used to justify a multi-million dollar software initiative for firms with 500 or more software engineers.

## Call to Action

I encourage those of you with solid numbers to throw stones. One of my goals in writing this paper is getting the community to think about, discuss, and share like numbers. If you have more concrete numbers, I encourage you to put them into the public domain. Challenge my assumptions and summarize your experience so the community can use it. However, if you do, please tell the community how you derived the numbers, what their source is, and how you normalized them. Do not just throw

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

stones at my results. That would be counter productive. Share your experience with others and help the community to develop realistic benchmarks that they can use for comparison purposes.

I really would encourage those who do not have numbers to develop them. Throughout my entire career, I have used the numbers to win the schedule and budget battles, to acquire investment dollars, to improve my decision-making abilities, and most importantly to win management's trust. I gained credibility with management at all levels of the enterprise by discussing both the technical and business issues associated with my proposals. I was successful in getting approvals because I emphasized business goals and showed management that what I was proposing made good business and technical sense. It is not surprising that I am a strong advocate of managing by the numbers. Try it, and I think you will like it.◆

## References

1. Reifer, D. J. Making the Software Business Case: Improvement by the Numbers. Addison-Wesley, 2001.
2. Reifer, D. J., J. Craver, M. Ellis, and D. Strickland, eds. "Is Ada Dead or Alive Within the Weapons System World?" CROSSTALK Dec. 2000: 22-24.
3. Ada and C++: A Business Case Analysis. U. S. Air Force, 1991.
4. Jensen, R. "Software Estimating Model Calibration." CROSSTALK July 2001: 13-18.
5. Reifer, D. J. "Comparative Accuracy Analysis of Cost Models to Activity-Based Costing for Large Scale Software Projects." Reifer Consultants, Inc., 1996.
6. Ferens, E., and D. Christensen, eds. Calibrating Software Cost Models to Department of Defense Databases – A Review of Ten Studies. Air Force Research Laboratories, Feb. 1998.
7. Florac, W. A., and A. D. Carleton, eds. Measuring the Software Process. Addison-Wesley, 1999.
8. Brooks, F. The Mythical Man-Month, Anniversary Edition. Addison Wesley, 1995.
9. Boehm, B. W., C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, eds. Software Cost Estimation with COCOMO II. Prentice-Hall, 2000.
10. Kruchten, P. The Rational Unified Process. Addison-Wesley, 1998.
11. Royce, W. Software Project Management: A Unified Framework. Addison-Wesley, 1998.
12. Reifer, D. J. A Poor Man's Guide to Estimating Software Costs. 8th ed., Reifer Consultants, Inc., 2000.
13. Paulk, M. C., C. V. Weber, B. Curtis, and M. B. Chrissis, eds. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, 1995.
14. Clark, B. "Quantifying the Effects on Effort of Process Improvement." IEEE Software Nov./Dec. 2000: 65-70.

## About the Author

**Donald J. Reifer** is one of the leading figures in the fields of software engineering and management, with more than 30 years of progressive experience in government and industry. In that time, he has served as chief of the Ada Joint Program Office and the director of the Department of Defense Software Reuse Initiative. He is currently the president of Reifer Consultants, Inc., which specializes in helping clients improve the way they do business. Reifer's many honors include the American Institute of Aeronautics and Astronautics Software Engineering Award, the Secretary of Defense's Medal for Outstanding Public Service, the NASA Distinguished Service Medal, the Frieman Award, and the Hughes Aircraft Fellowship. Reifer has more than 100 publications, including *Software Management Tutorial* (6th edition) and *Making the Software Business Case: Improvement by the Numbers.*

**P.O. Box 4046**
**Torrance, CA 90505**
**Phone: (310) 530-4493**
**Fax: (310) 530-4297**
**E-mail: d.reifer@ieee.org**

## WEB SITES

### Earned Value Management

www.acq.osd.mil/pm

Sponsored by the Office of the Under Secretary of Defense (Acquisition, Technology, and Logistics) Acquisition Resources and Analysis/Acquisition Management, the Earned Value Management Web site provides information on earned value project management (EVM) for government, industry, and academic users. Find current editions of policy documents, information on government EVM contacts and International Performance Management Council members, speeches, training material, EVM software, supplier links, frequently asked questions, and more.

### Project Management Institute

www.pmi.org

The Project Management Institute (PMI) claims to be the world's leading not-for-profit project management professional association, with more than 86,000 members worldwide. PMI establishes project management standards, provides seminars, educational programs, and professional certification for project leaders.

### Defense Contract Management Agency

www.dcma.mil

The Defense Contract Management Agency is the Department of Defense contract manager, responsible for ensuring federal acquisition programs, supplies, and services are delivered on time, within cost, and meet performance requirements. The agency is currently conducting in-plant observations using the Capability Maturity Model® to deploy a standard methodology via continuous process evaluations of contractors. Details concerning the process, responsibilities, and outcomes are captured in the "Method Description Document" available on the Web site.

### Software Technology Support Center

www.stsc.hill.af.mil

The Software Technology Support Center (STSC) is an Air Force organization established to help other U.S. government organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.