migrating from assembly languages to higher order languages. It is now a major stimulus for developing and using very high-level languages and fourth-generation languages to reduce software costs.

5. Variations between people account for the biggest differences in software productivity.

Studies of large projects have shown that 90th-percentile teams of software people typically outproduce 15th-percentile teams by factors of four to five. Studies of individual programmers have shown productivity ranges of up to 26:1. The moral: Devote everything you can to get the best people working on your project.

6. The overall ratio of computer software to hardware costs has gone from 15:85 in 1955 to 85:15 in 1985, and it is still growing.

This relationship has done more than anything else to focus management attention and resources on improving the software process.

7. Only about 15 percent of software product-development effort is devoted to programming.

In the early days, there was a 40-20-40 rule: 40 percent of the development effort for analysis and design, 20 percent for programming, and 40 percent for integration and test. Now, the best project practices achieve a 60-15-25 distribution. Overall, this relationship has been very effective in getting industrial practice to treat software product development as more than just programming.

8. Software systems and software products each typically cost three times as much per instruction to fully develop as does an individual software program. Software-system products cost nine times as much.

A software system contains many software modules written by different people. A software-system product is such a system that is released for external use. The discovery of this cost-tripling relationship has saved many people from unrealistically extrapolating their personal programming productivity experience into unachievable budgets and schedules for software-system products.

9. Walkthroughs catch 60 percent of the errors.

The structured walkthrough (software inspection) has been the most cost-effective technique to date for eliminating software errors. It also has significant side benefits in team building and in ensuring backup knowledge if a designer or programmer leaves the project.

I had a hard time picking number 10. I ended up with a composite choice:

10. Many software phenomena follow a Pareto distribution: 80 percent of the contribution comes from 20 percent of the contributors.

Knowing this can help a project focus on the 20 percent of the subset that provides 80 percent of the value for improvement. Some examples:
- 20 percent of the modules contribute 80 percent of the cost,
- 20 percent of the modules contribute 80 percent of the errors (not necessarily the same one),
- 20 percent of the errors consume 80 percent of the cost to fix,
- 20 percent of the modules consume 80 percent of the execution time, and
- 20 percent of the tools experience 80 percent of the tool usage.

I think it has been a strong credit to the software metrics field that it has been able to determine and corroborate these and many other useful software metric relationships. And there are many useful new ones coming along. I look forward to reading about them in this department.

**SOFTWARE LETTERS**

Continued from p. 5

Science and Engineering, and research in our field is severely underfunded.

Although a liberal citation policy cannot solve such broad problems by itself, encouraging authors to cite related, historically significant research will help break down the narrow perspectives to which many of us are bound. In addition, it will encourage scholarship and intellectual honesty among authors. Also, having comprehensive bibliographies will greatly help those who want to learn more about the subject in an article.

It seems to me that increasing IEEE Software's citation limit could yield significant benefits to authors, readers, and our field as a whole.

Joseph A. Giguen
Program Manager
Computer Science Laboratory
SRI International
Menlo Park, Calif.

**Comments misread**

To the editor:

Douglas Schuler ("Real-World Coverage," Letters, July, p. 5) has misread my comment on SDI coverage ("Wasted Space?" Letters, May, p. 3). I merely posed that the space normally devoted to the umpteenth rehash of arguments, whether pro or con, be severely reduced. Never did I ask that "topics be excluded" or suggest that anyone "disregard real-world implications."

My point was that these important issues could indeed be covered, minus the customary hot air, in about a third the space normally used. A perusal of the letter from Frederick Seitz ("SDI Advocate's View," Letters, July, p. 5) confirms this figure.

If I exist "in a vacuum," that analogy should be extended to say that many others have their heads in pressure cookers — they are always steamed and can't seem to get out anything intelligible.

As soon as new information on SDI is reported, I will be the first to acknowledge its newsworthiness.

John D. Wolf
Programmer
SCS, Inc.
Fort Wayne, Ind.

We welcome your letters. Send them to Letters Editor, IEEE Software, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720. All submissions are subject to editing for style, length, and clarity.

---

**Share your experience**

Whether your experience tends to confirm or refute the opinions expressed here, we encourage you to share that experience with us. We welcome short submissions (less than 750 words) that cover practical experiences, factors influencing productivity and quality, attempts to overcome some of the weaknesses in certain metrics, risks in using imperfect measures, and the like.

We welcome both specific and general discussions of productivity and quality.

September 1987