Comment on "Capital-Intensive Software Technology"

Peter Wegner's mega-article, "Capital-Intensive Software Technology" (IEEE Software, July 1984, pp. 7-45), is a masterful survey of the local landscape called "software development," but I believe the article is based on an entirely incorrect assumption. He states, "Intuition suggests that a production process is capital-intensive if it requires expensive tools or if it involves large startup expenditures. Software development is becoming increasingly capital-intensive: its tools are becoming more powerful and expensive, and it requires greater early investment to reduce late expenditures." The article points out many interesting side-effects due mainly to the pervasiveness of personal computers, but I believe it fails to link these interesting side effects to anything remotely connected to "capital-intensive software technology."

First, software development is becoming increasingly cheap. Computers are nearly free compared to 10 years ago; compilers cost $395 instead of $20,000; editors, operating systems, and application generators are perhaps two orders of magnitude less expensive than they were a few short years ago.

Second, knowledge is tremendously undervalued. A BS in computer science can be purchased for a few hundred barrels of oil, the profit reaped from a few acres of farmland, the cost of a new car, or the cost of a European vacation for two. The cost of software development has fallen to the point where thousands of cottage industry programmers have glutted the software market with programs of every description. They literally give away their expertise in every field of endeavor from stock market analysis to electronic circuit design. The problem, I believe, is that software development simply costs more than hardware development. This is due to the steep decline in hardware costs—even steeper than the decline in software costs—rather than any comparative surge in software development costs. The amount of advertising and promotional dollars needed to bring out a new product in an overcrowded market is the culprit.

Software development cannot be cast into the mold of capital-intensive manufacturing or any other economic artifice because knowledge and creativity defy economics. One good idea is worth all of the software development tools and methodologies combined.

Do not get me wrong; I am an advocate of better tools and more productive methodologies, but it is a mistake to believe these are a panacea for that ail software technology.

Ted G. Lewis
Department of Computer Science
Oregon State University
Corvallis, OR 97331

Author's reply

Ted Lewis overlooks the fact that techniques for constructing small programs do not scale up to large programs. This was realized by the computing profession in the late 1960's when the discipline of software engineering was born. Software engineering's primary objective was to develop a capital-intensive framework for software production that would parallel capital-intensive technology for building large physical structures.

Tony Hoare, in his article, "Programming: Sorcery or Science" (IEEE Software, April 1984, pp. 5-16), suggests that small programs are like sandcastles while large programs are like city blocks. His assertion that one would not use even a prizewinning builder of sandcastles to build a city block nicely captures the idea that techniques for building small programs do not scale up to large programs. Lewis appears to suggest that all programming is simply the building of sandcastles and that city blocks do not exist. This is manifestly not the case.

Capital-intensive technology is required both when a software product is too complex to be managed by a small number of individuals and when a simple software product is mass-produced for a large market. Software technology has made Cobol compilers, spreadsheets, and Pac-man affordable by distributing their capital development costs over many consumers, just as industrial technology has made automobiles, television sets, and personal computers affordable.

Software products differ from industrial products in that copies of a software product require few raw materials and little labor to produce. But this is balanced by the fact that widely distributed software products incur support and maintenance costs comparable to the production costs of an industrial product. The initial versions of Unix and Lotus 1-2-3 were produced on shoe-string budgets, but their widely distributed successor systems, Unix 4.2 and Symphony, require capital-intensive organizations to support their distribution. Thus, savings in costs of mass production are balanced by expenses incurred by mass distribution, and the overall economics of mass-producing software products is therefore not too different from that of mass-producing industrial products.

The current software boom has spawned many small start-up companies to manufacture specific systems in areas such as automatic typesetting, computer-aided design, and computer-aided instruction. These companies are heavily capital-intensive, typically requiring an initial investment of $1 million to $5 million and an elapsed time of at least two years before their first commercial product is sold.

Large, evolving software systems that control banks, airplanes, and ships are even more capital-intensive. They are generally constructed either by very large (Fortune 500) companies or by the organization they are designed to control. They require careful planning over a long time horizon and must take into account post-