December 2001

No Silver Bullet? Reducing the Demand for IT Labor

Jeffrey Merhout
Virginia Commonwealth University

NO SILVER BULLET? REDUCING THE DEMAND FOR IT LABOR

Jeffrey W. Merhout
Virginia Commonwealth University
jmerhout@acm.org

Abstract

In the United States, there is currently an acute shortage of qualified information technology professionals. While there are no simple solutions to this dilemma, this conceptual paper explores the possibility that a robust software component market might be a silver bullet that could dramatically alleviate this shortage. A functional component market based on a pay-per-use system would lead to more labor specialization, thereby increasing the productivity of the average software developer. In the long run, more software applications could be built using fewer labor resources.

Keywords: Software development, software engineering economics, software component markets

Introduction

The media has highly publicized the current acute shortage of qualified information technology (IT) professionals in the United States. Estimates of the number of unfilled jobs vary widely, but tend to range from 350 to 500 thousand nationwide. As evidence of this enormous demand, one only has to review the starting salaries of entry-level IT positions that require only an undergraduate degree. A new college graduate can reasonably expect to earn between $45 thousand and $70 thousand per year, and some recent graduates with only about two years of expertise in very specialized applications (e.g., ERP) can possibly make over $100,000 per year!

Although many have suggested ways to increase the supply of qualified professionals, the primary purpose of this conceptual research is to examine the other side of the economic equation, the demand-side. As noted by Kemerer (1998), the demand-side focuses on effectively using information technology. Kemerer further comments that the supply-side has received more attention in the area of software engineering economics (SEE), which is the application of economic principles to software development. This statement is supported by the dearth of articles on this topic in the literature. History suggests that a demand-side review of supply-demand inequities, specifically as they relate to labor markets, is useful to both the academic and business communities, especially in focusing on ways to decrease the demand. At the very least, a dialogue on this topic is warranted as many communities and academic institutions continue to make major investments to supply the IT labor market.

The next section of this paper will provide an extensive background on labor markets, labor productivity and the increasing standards of living realized as labor resources are used more efficiently. This will be followed by a brief description of the term "silver bullet." Thereafter, a detailed discussion of the promising possibilities of a software component market will ensue, including an argument that such a market is possibly the only true silver bullet available. The paper will conclude with a brief mention of other potential solutions. These include buying versus building applications, building only strategic applications, end-user programming and the need to develop outstanding software designers and developers.

Background

Although I primarily wish to explore the demand-side of the IT labor shortage, it may not always be clear whether a certain idea, method or technique is truly affecting the demand-side or the supply-side, or perhaps both. Kemerer (1998) describes the demand-side of SEE as focusing on effectively using information technology. Examples include "IT-induced changes to markets"

(Kemerer 1998, p. 63) such as "drivers of e-commerce" (Kemerer 1999). Therefore, if one assumes that the demand-side refers to the demand for software and IT applications, then we could never expect (or desire) to drive down this demand since it helps to fuel our national economy. However, we do need to try to reduce the demand for IT labor since the supply will most certainly never equal the demand. This paradox of the desire to decrease demand for the labor required to produce an ever-increasing demand for software products can only be resolved by increasing the productivity of each unit of labor input. The average IT worker needs to produce more and better quality software (the output) in a given period of time.

As espoused by Brynjolfson and Hitt (1998), productivity growth is eminently important since it is a key determinant of our standards of living and the wealth of our nation. According to Drucker (1992), productivity in the blue collar industries such as manufacturing, farming and construction increased by three to four percent annually during the last century. This enormous increase has led to higher living standards and quality of life in developed countries. However, this "productivity revolution" (Drucker 1992, p. 93) is essentially over because those who make and move things (blue collar workers) constitute only about 20% of the work force in developed countries, with this percentage continuing to decrease. Furthermore, the productivity of the work force taking the place of laborers, the knowledge worker and service worker, has heretofore shown no measurable, significant increase in productivity. If we can solve this productivity problem, we can reasonably expect that our standards of living will, at the least, be maintained. Indeed, Drucker (1999) calls knowledge worker productivity "the biggest of the 21st-century management challenges" (p. 157). Since software is perhaps the ultimate embodiment of knowledge (Baetjer 1998), one could argue that the IT worker, specifically the software developer, exemplifies the knowledge worker. If we can find ways to increase the productivity of this worker, we can both ease the shortage of IT labor and help to maintain or increase our standards of living.

In order to fully appreciate the potential power of productivity growth, one need only review the history of the American industrial economy. The Industrial Revolution transformed the world from an agrarian to an industrial society. Eli Whitney is credited with demonstrating that interchangeable parts could be successfully used in the manufacture of firearms (Davidow and Malone 1992; North 1981), thereby helping to move industry from a craft society to higher volume production processes. In 1895, Frederic Taylor began studying what was later called scientific management (Chandler 1977). Taylor analyzed the tasks of manual workers and broke them down into basic procedures to identify wasted steps (Drucker 1999). He advocated making each worker a specialist in a single job, an interchangeable cog in an efficient production process. In addition, the tools and equipment required to assist the worker were analyzed and redesigned to allow the worker to accomplish the task as quickly and effortlessly as possible. Both Chandler and Drucker note that, while Taylor's ideas had limitations and numerous critics, many of his concepts were adopted in American factories. Indeed, Drucker claims that Taylor has had more of an impact on American industry than any other person. Furthermore, his ideas were incorporated by the American war effort in World War II, allowing the United States to outnumber the enemy on the battlefield while still out-producing them in the factories.

In the early 1900's, Henry Ford applied Taylor's division of labor ideas to perfect the mass production process in automobile manufacturing (Chandler 1977; Drucker 1999). By utilizing a carefully designed moving assembly line and assigning each worker a highly specialized job, Ford was able to reduce the amount of labor hours required to make a Model T from over 12 hours to about 1.5 hours. This reduction in labor cost allowed Ford to reduce the price of this automobile from more than $3,000 to less than $1,000 (Mansfield 1999), or about half the price of the nearest competitor, while paying the highest non-skilled labor wages in the country (Chandler 1977).

Increasing IT Labor Productivity

Clearly, this notion of a division of labor into specialized tasks was successful in the automobile and other manufacturing industries. However, the information systems community is typically still building systems and applications one at a time, much in the same way automobiles were being built at the turn of the last century. Ford realized that a method of production that relied heavily on skilled craftsmen would never produce affordable automobiles. In order to reduce labor costs and make the cars affordable enough for the common person, Ford moved to mass production where the productivity of the average worker was increased dramatically. One could argue that the information technology field is in a similar position as the automobile industry was a century ago and is ripe for dramatic changes in the way systems are built.

A detailed review of the literature has uncovered several promising ideas to increase productivity, even if they all do not warrant the classification of a true software development "silver bullet" as discussed by Fred Brooks (1987). Brooks utilizes this metaphor of the silver bullet (used to slay werewolves in folklore) to describe how the software industry desires to achieve an order of magnitude increase in productivity and reliability. The goal is to slay the "monster of missed schedules, blown budgets, and flawed products" (Brooks 1987, p. 2). Brooks believes that software is inherently complex, perhaps more than any other construct built by humans, and that this complexity makes it unlikely that a silver bullet will ever be discovered. However, as Brooks notes

about software engineering, and as more optimistically believed by this author and by Cox (1990), there are several things that could be done (both on the demand and supply sides) to advance the field, thereby helping to alleviate the severe labor shortage.

Software Component Market

Perhaps the most promising possibility for a silver bullet would be a true economic market for reusable software components (Baetjer 1998; Cox 1990; Lewis and Oman 1990). Indeed, Baetjer claims that such a market is "what the industry in general needs in order to take a quantum leap forward in productivity" (p. viii). Since such a market would have to be electronically based and facilitated through information technology (similar to e-commerce), it seems reasonable to call it a demand-side solution in the spirit of Kemerer's description (1998). Although a market such as this likely would increase the overall demand for applications, in theory it would eventually lead to a decrease in the total number of developers required to satisfy this demand. Cox (1990) calls for a software industrial revolution to pull software developers out of the software crisis, a crisis where software is too expensive and time-consuming and often possesses less than ideal levels of quality. Cox uses the analogy of the industrial revolution to suggest that the software industry must evolve away from building applications like the gunsmith built guns in colonial times. This type of craftsmanship needs to be replaced by constructing applications from interchangeable parts called software components and sub-components, available in a component market. Even Brooks (1995) "enthusiastically agree(s)" with Cox's call for a "reusable, interchangeable component approach" (p. 210).

As discussed by Szyperski (1998), "software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system. Composite systems composed of software components are called component software" (p. 3). Cox (1995) discusses the reasons why a comprehensive component market does not currently exist. Since tangible goods consist of atoms, they are easy to possess, buy and sell, but difficult to replicate and distribute. On the other hand, electronic goods consist of bits, are extremely easy to replicate and distribute, but suffer from the inherent properties of intangible goods, making them more difficult to possess, acquire or sell. Since traditional economic markets were created for tangible goods, they do not work well for electronic goods.

Economist Howard Baetjer (1998) extends Cox's call for markets to support software components. Other industries generally rely heavily upon specialization and division of labor in an extended chain of production where specialists build the components and sub-components of most products. However, most software is still built from scratch. Tangible goods producers rely upon the market to provide prices and information on supply and demand, thereby allowing them to decide what kind of products to produce and in what quantities. Other than a few class libraries (preprogrammed code), there is a limited supply of software components or objects to purchase. Moreover, in the situation of libraries, the market does not work the way it does with tangible products because one must purchase the entire library in order to get one particular object. As Baetjer (1998, p. 25) elucidates,

> in this limited market state for software components, there is little incentive for firms to specialize in producing certain types of inputs, and little price guidance to tell them either what to specialize in or what sub-components it would be economical for them to incorporate. Accordingly, the structure of software production stays short, the division of knowledge stays limited, and the industry's productivity stays low.

In order for industrial marketplaces to function correctly, producers must be compensated for the goods they sell (Baetjer 1998). This is a rather mundane function when the goods are tangible; each instance represents a unit that can be touched and counted. The market has evolved to easily facilitate the physical transfer of goods and to accommodate payment for each instance of the product exchanging hands. Software, on the other hand, is sold by the copy, not by the instance of usage (Cox 1992). Unfortunately, it is very easy to make and distribute copies of the software without deriving additional income for the producer. Although there are copyright laws and licensing agreements designed to protect the developer, these systems do not function very well. As explained by Baetjer (1998, p.128),

> in the absence of better means of making sure that they will be paid for what they produce, software developers simply do not go into the business of producing specialized, fine-grained components. From the standpoint of component users, this arrangement is expensive and troublesome in that they must often buy more than they need and spend time and effort staying in compliance with royalty and licensing provisions.

Baetjer and Cox advocate a charge or pay-per-use system that requires payment for each instance of software use rather than for each copy. They suggest giving away the software at no cost and deriving all revenue from usage. Obviously, such a system based on usage would require a special technical infrastructure to facilitate accounting requirements. One such system, called superdistribution, is described in detail by Mori and Kawahara (1990). Although these authors present technical details beyond the scope of this current paper, the basic premise is that the superdistribution architecture requires hardware that collects and

accumulates accounting information based on usage. Periodically, this revenue information is sent to a clearinghouse that debits and credits each software producer (in the entire chain of production) and user accordingly (Cox 1992). It is interesting to note that organizations are marketing services today on the Internet that appear to serve as this clearinghouse function (e.g., Superdistribution™, Inc.).

These Internet entrepreneurs suggest that this is an idea whose time may have come. However, as Baetjer (1998) and Cox (1996) note, this concept is more of a paradigm shift than a technological one. As such, there are many social, cultural and institutional hurdles to overcome as the component market develops and evolves. One such issue involves the current developer mindset to build software from scratch (Baetjer 1998). Current and future software developers and engineers need to adopt the philosophies of object-oriented technologies, particularly reuse (both as a user and as a producer) and of small-granular components (Cox 1997). While there will be many setbacks and mistakes made along the way, the market will provide valuable feedback that will force the developers to adapt and evolve, to learn, to specialize and to become markedly more productive (Baetjer 1998).

Demand-Side Solutions

Although the development of a market for software components appears to be the most promising silver bullet on the horizon, there are other ideas and technologies that could, collectively, ease the demand for IT labor. Perhaps the easiest way to decrease the demand for IT labor is to avoid building applications when at all possible. In 1987, Brooks advocated the development of mass markets for software applications (buying rather than building) as a very promising way of dealing with some of the issues of software development. This is often called shrink-wrapped or commercial off-the-shelf (COTS) software. Andriole (1990) remarked that, far too often, organizations fail to utilize off-the-shelf products even when they are readily available in the marketplace. In his reflective essays assessing the ideas of his prior classic publications, Brooks (1995), noted that his call for shrink-wrapped software had been realized by a growing industry. He extends this idea further by calling for organizations to use commercial products such as databases and spreadsheets as components for other customized applications. This would appear to be the ultimate example of the power of the component market as the economies of scale realized in the commercial product would result in the cost of a new application being dramatically less than if built from scratch. A hardware analogy might be when the automobile manufacturer embeds a purchased microchip into their engines rather than building their own computer to use in the auto.

Other demand-side strategies could conceivably ease the IT labor shortage, at least to a certain degree. These include building only strategic systems, end-user programming, reducing maintenance requirements, using open systems, and automatic programming facilitated by artificial and expert system technology. However, none of these approaches promises to be a silver bullet, and for the sake of brevity, will not be discussed further in this paper. Instead, I will turn the focus to a supply-side strategy that deserves attention by both the academic and practitioner communities. Kemerer (1998) defines the supply-side of software economics as topics "which focus on the efficient provision of IT resources" (p. 63).

One could certainly argue that the single most important IT resource is people. Indeed, Brooks (1987) discusses "growing" great designers as a promising strategy to deal with the inherent complexity of software and notes that constructing software requires a creative process. Furthermore, Blackburn et al. (1996) suggest that "creative people are the real silver bullet" (p. 884). Brooks (1995) adds that the quality of the people on the project are more important than tools or techniques and praises research that is increasingly paying more attention to the human resource side of development issues. While it is encouraging that the human component is becoming more of a focus in academia, there is still much to be accomplished. Practitioners are urging universities to improve their curricula by including more state of the art software engineering courses such as quality control, change management, and project planning and management (Jones 1999). In addition, academic scholars note the interdisciplinary nature of software engineering (Kemerer 1998) and call for an increased emphasis on multidisciplinary education and research (Andriole 1990). In short, an increased focus on the investment in human capital by software developers (a supply-side solution) combined with the development of effective markets for software components could potentially go a long way to alleviate the chronic IT labor shortage.

Conclusion

The main premise of this paper is that the software industry has a labor shortage situation that appears to be nearing crisis stage and could continue to worsen until some rather dramatic institutional changes are made. The most promising potential solution, based on the software development and SEE literature, is the creation of a true economic market for software components. Such a market would conceivably lead to an increase in IT labor productivity by allowing specialization and division of labor. Increasing the productivity of knowledge workers, such as software developers, would maintain or raise the standards of living

in developed nations. Perhaps more important to information systems researchers is the argument that that such a market would facilitate the development of higher quality software applications while utilizing fewer labor resources. However, one must realize that software components are unlike tangible goods and will require special market mechanisms. A software component market is by no means a simple solution nor is it guaranteed to immediately operate effectively or efficiently. However, both researchers and practitioners have a stake in this issue and should, at a minimum, consider and discuss these concepts since something needs to be done before the IT labor shortage reaches a real crisis level. Furthermore, future research needs to identify existing component markets and empirically test whether software development productivity increases in such a scenario.

References

Andriole, S.J. *Getting It Right!: Information System Design Principles for the 90s,* AFCEA International Press, Fairfax, VA, 1990.

Baetjer, H. *Software as Capital: An Economic Perspective on Software Engineering*, IEEE Computer Society, Los Alamitos, CA, 1998.

Blackburn, J.D., Scudder, G.D., and Van Wassenhove, L.N. "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers," *IEEE Transactions on Software Engineering* (22:12), December 1996, pp. 875-884.

Brooks, F.P. "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer* (20:4), April 1987, Reprinted in *Software Project Management: Readings and Cases*, C.F. Kemerer (ed.), Irwin, Chicago, 1997, pp. 2-14.

Brooks, F.P. *The Mythical Man-Month: Essays on Software Engineering,* Anniversary Edition, Addison-Wesley Publishing Company, Reading, MA, 1995.

Brynjolfsson, E., and Hitt, L.H. "Beyond the Productivity Paradox," *Communications of the ACM*, (41:8), August 1998, pp. 49-55.

Chandler, A.D. *The Visible Hand: The Managerial Revolution in American Business*, The Belknap Press, Cambridge, MA, 1977.

Cox, B.J. "Planning the Software Industrial Revolution," *IEEE Software*, (7:6), November 1990, Reprinted in *Software Project Management: Readings and Cases*, C.F. Kemerer (ed.), Irwin, Chicago, 1997, pp. 401-414.

Cox, B.J. "What if There Is a Silver Bullet?" *Journal of Object-Oriented Programming* (5), June 1992, pp. 8-9, 76.

Cox, B.J. *Superdistribution: Objects As Property on the Electronic Frontier,* Addison-Wesley Publishing Company, Reading, MA, 1996.

Cox, B.J. "Superdistribution and the Economics of Bits," *IEEE Software*, (14:1), January 1997, pp. 22-24.

Davidow, W. H., and Malone, M.S. *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century*, HarperBusiness, New York, 1992.

Drucker, P.F. *Managing for the Future: 1990s and Beyond*, Truman Talley Books, New York, 1992.

Drucker, P.F. *Management Challenges for the 21st Century,* HarperBusiness, New York, 1999.

Jones, C. "The Euro, Y2K, and the US Software Labor Shortage," *IEEE Software,* (16:3), May/June 1999, pp. 55-61.

Kemerer, C.F. Personal e-mail conversation, November 1999.

Kemerer, C.F. "Progress, Obstacles, and Opportunities in Software Engineering Economics," *Communications of the ACM,* (41:8), August 1998, pp. 63-66.

Lewis, T.G., and Oman, P.W. "The Challenge of Software Development," *IEEE Software,* (7:6), November 1990, pp. 9-12.

Mansfield, E. *Managerial Economics: Theory, Applications, and Cases*, Fourth Edition, W.W. Norton & Company, New York, 1999.

Mori, R., and Kawahara, M. "Superdistribution: The Concept and the Architecture," *Transactions of the IEICE,* (E73:7), July 1990, pp.1133-1146.

North, D.C. *Structure and Change in Economic History,* W.W. Norton & Company, Inc., New York, 1981.

Szyperski, C. *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, Harlow, England, 1998.

Superdistribution™, Inc., Available online at <www.superdistributed.com>, 2000.