How Will the Software Talent Shortage End?[1]

Avron Barr and Shirley Tessler
SCIP Software Industry Study[2]

Projections from all sources show a widening gap between supply and demand for software professionals. This shortage is worldwide, not just in the US, is not limited to any specific technology, like Java, SAP or Year 2000 workers, and extends across all industries, not just high-tech. A number of people we've talked to about the current shortage in skilled software labor argue that, while there may be short-term tightness in labor markets, these bottlenecks can't last very long in a free economy: salary adjustments and other market forces will naturally balance the system by attracting more people into the field. Indeed, many point to historical oversupplies of some scientific and engineering disciplines as a caution against pleas to introduce programs designed to increase the number of software developers. While we can't predict the future, we feel that a sustained software talent shortage would have serious ramifications. We hope our thoughts on the issue will stimulate a fruitful discussion.

All agree that there are only three ways this workforce shortage can end:

1. The supply of talented labor rapidly increases to catch up with rising demand, as a result of higher salaries, more new high-tech graduates, etc.;
2. New software development technologies are introduced which increase the productivity of the existing labor force; or
3. Demand for software slacks off and the slowly growing labor pool eventually catches up.

Economists, technologists and other optimists are certain that either solution #1 or solution #2 will solve the talent shortage problem in short order. In our opinion, damage is already being done: serious project risks presented by prolonged talent shortages lasting a decade or more will eventually cause a decrease in demand for new systems to levels that can be reliably produced by existing talent supplies, which will continue to grow slowly.

The Blight at the End of the Tunnel

The shortage of software talent is caused, fundamentally, by increasing demand for software of all sorts: products from software publishers like Microsoft, Oracle and Sega (growing at 12-15%

[1] This is a draft circulated for comment. The article will appear in *American Programmer*, January, 1998, Cutter Information Systems, Arlington, Massachusetts.

[2] All SCIP papers are available on our website at http://www-scip.stanford.edu/scip/

each year), corporate information systems from the back office to e-commerce (growing at 10-12%), and the software that is embedded in increasing amounts into all kinds of products from automobiles to cellular phones to toys (for which there is no data on current size or growth). The broad use of increasingly complex software across so many industry sectors is both the reason for the shortage and the cause for concern — software has become critical to the economy.

Technologists and policy makers point to erroneous past projections of labor shortages in new technologies, such as the projection that telephone use would grow so large that every person in the US would need to be employed as a telephone operator. They use the telephone operator metaphor to argue that some imminent technical breakthrough will surely reduce the need for software specialists of all sorts. While it is true that new software development technologies will eventually appear, there are currently no new tools or techniques on the horizon which will significantly impact the productivity of software developers across broad classes of software. Furthermore, technology adoption is slow enough that even an easy to implement and integrate invention takes many years to penetrate even a small portion of the software industry. So, we are still left with a straight-forward supply and demand situation for software.

We should point out that we cannot be very specific about what kind of software workers are in most critically short supply. We believe that there is less of a problem finding "entry level" programmers. But the talent involved in successful software projects is not just entry-level people. It includes systems analysts, systems architects, several types of designers, programmers of all sorts, testers, documentation writers and even trainers. Even more importantly, it includes project management and a number of other technical management specialties. We know that for all skills, the most talented are much more productive and less error prone that the average (a factor of 20 is often cited), and that experience and familiarity with the system they are working on increases productivity for all levels of talent in any of the jobs. But we don't know where the talent shortage is most critical, what skills are in most demand, or how entry-level workers match up with the talent levels and specific skills in short supply.

Some economists and government policy makers feel that increasing the supply of software people, by raising salaries and funding new educational programs, is the way to deal with the situation. After all, they argue, many current programmers had no formal training in computing and migrated to the profession from engineering or business or other fields. But software development has changed over the years: it is as complicated as any engineering task, and most people with a talent for software are already doing it. Moreover, demand is growing fast, perhaps two hundred thousand unfilled new jobs a year in the US, and increasing the supply of skilled workers is a slow process. Before significant progress can be made in expanding the trained labor force, many open positions will be filled with inexperienced people who have, on the average, lower productivity and possibly less training or talent in the various skill areas.

This "deskilling" of the software talent pool will eventually slow down the demand for software. Reduced demand seems like an unlikely prediction at a time when everyone owns a computer, every household device and toy has a chip in it, and corporations compete on the basis of Internet marketing schemes, supply chain management and database marketing. But as software talent becomes scarce, systems built by less-experienced or less-trained teams may result in even more serious losses for many organizations than they already experience — delayed projects (like airports), abandoned projects (like the IRS revamp), and poor-quality systems (like air traffic control). Disenchantment and pragmatic reassessment of how many systems their people can realistically deploy will make CEO’s, CIO's and product designers re-think their investments in new projects. This situation is bad news for the software industry and for high-tech generally. Furthermore, because so much of our national economic advantage is based on the rate of innovation we can achieve, slowing down the software race is bad for the economy generally.

Expanding the Supply of Software Talent

Of course, if we could quickly increase the supply of talent, we could eliminate this labor shortage in short order. Indeed, if anyone off the street could become a programmer tomorrow, there would be no shortage (and no welfare system). How much can the talent pool be expanded by education, immigration, and cross-over of intelligent people from other fields into software?

We believe that demand is growing much faster that the educational capacity of current institutions, which are disinclined to rapidly grow their capacity. (We have recommended the establishment of new, industry-funded trade schools, but see no likelihood that this will happen in the next decade.) Immigration in the current political climate is not a solution, and this shortage is a global one, in any case.

Since entry-level people seem to be easier to find, some believe that software shops that insist on certain specific skills or experience levels are being too selective in their hiring practices. But talented programmers are much more productive than average developers, and are rare. Furthermore, companies are reluctant to put inexperienced people onto mission critical projects — they can slow things down and are much more likely to make costly errors.

So it boils down to how many people already employed elsewhere will be attracted into software and how those people who do cross over, e.g., because the salaries are rising, will impact the talent pool. Will they be, on the average, as productive as the current workforce? Not everyone has an aptitude for the various skills required in software: architecture, design, coding, debugging, project management, etc. Furthermore, developers at the low end of the talent curve can have a negative impact on team productivity and software quality.

In our opinion, even with substantial salary increases, software jobs will not attract an adequate number of people from other disciplines with real talent for the various development tasks:

- There is a lot to learn before they can even start to contribute. Software development is an increasingly complex activity involving tools that take even the brightest, most talented people months to master. Software designers and architects, higher-paid and higher-prestige jobs indeed, must be familiar with a range of technologies and existing systems that takes years. (A bad designer or architect can nullify the work of a hundred great programmers.)

- Software development for the most part is still not a high-prestige position in most firms or in society at large, although that is changing slowly. Even amongst engineers (who are the most likely cross-over group), software people are looked down on as “not real engineers” because of the lack of maturity of the field as well as the historically low esteem of the software parts of engineering teams.
- Most software development work is not as attractive as it might first seem. It is characterized by tedious tasks and constant time pressure. For most software people, the opportunity to do the “creative” parts of their jobs comes along rarely compared to the typical situation of working out the bugs in the code against a looming deadline.
- The work is intolerant of errors, like doctors and airline pilots, but without the perks. Both conceptual and detailed aspects of the work must be perfect: all i’s dotted and t’s crossed.
- Most programmers are rarely appreciated for their work, except by their colleagues, and often chastised by management for their errors or their tardiness.
- Programming is not people work, not outdoor work, and not for everybody. It has attracted over the years a certain personality type, which in turn has kept others away from the field. The reluctance of many young people to consider a career in computing, because of the nature of the people who do, has been called “The Dilbert Syndrome.”
- The demographics don’t look good. Future growth of the labor pool will be slowed by the retiring of the first generation of programmers. Changing workforce demographics, especially the increased percentage of women and minorities, is also problematic since these groups are historically less inclined than white males to enter engineering disciplines.

We are left with a situation in which the labor pool is expanding much more slowly than demand is growing. Firms are trying to find the talent they need for current projects by recruiting it away from other firms. To understand how this dynamic is evolving, it is best to segment the industry.

Segmenting the Software Industry into Three Tiers

Software is used in many ways to create business value. In some sectors of the software industry, market forces are clearly already at work. Rising salaries are attracting top talent to meet rising production demand: talent from other sectors, from abroad, and from schools.

At the top tier of software organizations are the VC-funded software startups, the “boutique” software services firms, and the software publishing houses. In this segment, where growth is nearly 15%, the operations of natural labor market forces is quite apparent. There are three things to note about this tier of the software industry:

- These firms were the last to feel pinch. Top salaries, opportunity to own equity, and stimulating work have drawn the best software talent from other parts of the industry and an increasingly large portion of the best new graduates into these types of jobs. As a result, these firms did not feel any tightness in the labor market until recently. Therefore, dramatic salary increases, aggressive recruiting practices, and other market responses to the shortage are only beginning to appear there. We believe they will accelerate rapidly.
- It is apparent to CEO's and other executives in these firms that their survival depends on attracting top software talent. The need to produce innovative software of great complexity and high-quality as quickly as possible is a given — software is their bottom line.
- Furthermore, these companies are able to identify top talent and to offer the kinds of working environments that will attract the best software people. In fact, there is a radically different "perception" of software work in these firms from the working conditions in other companies that hire software talent.

Let's call these firms Tier 1 of the software industry. We believe this segment represents less that 10 percent of employment of professional software developers and produces only a fraction of the total business value attributable to software. Since growth in this tier has for several years involved migration of top talent from the rest of the software industry, we have found an increased awareness of the shortage elsewhere.

We divide the rest of the industry into two more tiers where the labor situation is quite different and where labor market forces have been much slower to operate. In Tier 2, which includes computer and other high-tech equipment manufacturers, communications companies and many firms in financial services and other IT-intensive industries, there is an growing realization that software development is critical to competitiveness. Some larger IT services firms would be classified as Tier 1, others would have to fall in Tier 2. In Tier 3, which includes most manufacturing and government organizations, people who have realized the strategic importance of software have no choice but to leave — there is no executive-level awareness here about the key role of software and no chance of responding to labor market changes even if software's value were clearly recognized.

Labor Market Forces in Tier 2

Since there is no ability or will to deal with the talent shortage in Tier 3, and the Tier 1 firms have mostly figured out what they need to do to recruit and retain top talent, all the market dynamics take place in Tier 2. The firms in this category are beginning to realize the critical role of software in their industries, but are still hobbled by old attitudes, practices and organizational structures. Their awareness of software's role in their business has moved beyond blaming software for delays in product launches and firing managers for failure to field new information systems. They realize that their products compete on the basis of software-enabled functionality and that their sustainable advantage in manufacturing, marketing, distribution or sales is increasingly dependent on systems that require advanced software capabilities. However, these firms are unable to respond to labor market tightness as fluidly as those in Tier 1:

- Using the argument that IT is not a core competency of the business, these firms responded to rising IT costs in the 1980's years by dramatically reducing their IT staffs, disbursing their talented software employees to other parts of the software industry, and in the process, sending a lingering message to those that stayed behind and to future recruits that this was not a place software talent was valued.

- Software jobs in these firms are not typically high-prestige positions. (This attitude may be changing, but only at "cultural" speed.) The organizational status of software people reflects the historical introduction of information technology. In IS, many software positions were recruited from the ranks of clerks when these organizations were formed in the 60's. In engineering groups, software people also often have low-status for many reasons. While some firms are able to offer higher salaries for small, specially-recruited teams of programmers, the majority of software-related positions are filled by whomever will accept the salaries offered.

- There are always people with credible software credentials who will take these Tier 2 jobs. Unfortunately, these firms have a hard time differentiating the truly talented from the pretenders. Aptitude tests are not politically correct. We contend that the most talented graduates, for example, are no longer even appearing in their recruitment process — they head directly for the high-paying and glamorous jobs in Tier 1. Managers with the technical background required to determine recruits' true talent have mostly left long ago for better positions. These firms are increasingly using IT services firms to outsource software development, but don't necessarily have the talent in-house to then manage an outsourced project to a successful conclusion.

- The acceptance of less experience or talent keeps salaries down, but makes it very hard to accomplish state-of-the-art systems, which in turn makes new recruits even less interested in signing up. While compensation is, of course, a factor in selecting a place to work, good software people also want to work with the newest technologies, produce innovative systems, and work with top-flight colleagues.

- Rather than change their practices to reflect the importance of software, these firms have tolerated massive losses — Standish Group data shows that 33% of projects are seriously delayed and/or over budget, and that another 40% of projects are abandoned completely.

As a result of these outdated attitudes and management practices, firms in Tier 2, where the talent shortage has been apparent from many years, are still losing talent, still not raising salaries and changing working conditions adequately to recruit against Tier 1, still not discriminating about whom they hire, and still not making the decisions required to turn this situation around, to retain the talent they have and recruit the talent they need.

In each industry, those few firms who invest in their software development capability will eventually prevail if they use this human resource well. The others will eventually stop investing, frustrated with their inability to field effective IT solutions and initiatives. The results for high-technology firms and for the economy in general will be serious. The results for American Programmers depend on where they are standing when the music stops.