Human Asset Management

Martin Allen
Independent Software Consultant

Widely recognized as the "father of modern economics," Adam Smith's seminal book, "The Wealth of Nations," included both tangible assets (machines, buildings, and land) and humans as essential wealth-generating resources. Our present high-technology industries are eager to invest in and protect their tangible assets (such as computer networks), but the modern accountancy paradigm forces the view of an employee as merely a cost. Humans are indeed primary assets, and this article provides guidance in assessing, utilizing, and enhancing their value.

Adam Smith was both influential and controversial in his inclusion of humans, or labor, as a major contribution to the wealth of a nation. Then again, Smith was not the only visionary in 18th century Scotland to acknowledge the human contribution to business and economics. With the Industrial Revolution in full flow, the common worker was perceived by industrialists as a replaceable cog in a machine, there to turn a profit and to be exploited. Cotton mill owner and social reformer Robert Owen would break that mold. On the banks of the River Clyde (near Glasgow) stands the one-time cotton mill town of New Lanark [1]. As testimony to its achievements, the mill operated from 1786 until 1968 and is now preserved as a World Heritage site. What distinguished the mill-owner from others at that time was the manner in which he nurtured the workforce: Having decent homes for workers, schools for their children, and cooperative shops delivering goods at a reasonable value contributed to the town's financial and communal success. For Owen, this was not philanthropy; it was the self-interest of capitalism, spiked with a social conscience. As clearly as the River Clyde was needed to turn the wheels of the mill machines, Owen recognized that people were needed to work the looms that weave the cloth. The end-result was employees who worked harder and better—and created a higher-quality final product.

Human Assets

Adam Smith and I share more than nationality; we both believe that humans are key assets in the pursuit of economically favorable results. For more years than I care to admit to, I have held a strong conviction that substantial and sustainable advances in industrial-scale software engineering will not come from new tools or programming languages; they will not emerge from the hyperbole surrounding heavyweight or lightweight processes; they will not come from accruing quality badges or collecting metrics; and they will not arrive via a modeling notation (such as the Unified Modeling Language). The potential for improvement is the most that various software tools, techniques, and initiatives can ever deliver. Substantive progress has been—and will continue to be—a direct consequence of employing software professionals and providing them with a suitable environment in which to operate. Competent personnel are an organization's pivotal assets.

"... people aspects seem to dominate the most expensive project disasters."

Having an appropriately trained software workforce has a double-edged effect. Educated teams focus on productive work; shared knowledge and experience give them the cohesion with which to address the intellectual tasks that comprise engineering. Equally significant is that teams spend less time evaluating and correcting substandard software artifacts simply because these are produced in negligible quantities by competent engineers.

While there is a widespread fallacy that technical issues are the primary source of project woes, people aspects seem to dominate the most expensive project disasters. Without a doubt, the majority of perceived difficulties are simply symptomatic of intrinsic people factors (*politics* is a popular euphemism for these). To quote from the influential book, "Peopleware":

> For the overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to explain the failure ... The major problems of our work are not so much technological as sociological in nature. [2]

Particularly in knowledge economies, the success or failure of technology programs hinges on assessing capabilities and recognizing the needs of the engineering teams. To this end, I have generated seven human-centric rules for software development organizations to adhere to:

- **Rule 1:** The main causal factors of project success, mediocrity, or failure should be recognized as human and organizational, not technological.
- **Rule 2:** Professionalism and software engineering competence should be assessed objectively and encouraged proactively by senior management.
- **Rule 3:** The number and seniority of software professionals employed within an organization should be commensurate with the magnitude and criticality of the required software systems.
- **Rule 4:** Organizations should provide an environment conducive to the intellectual task.
- **Rule 5:** Management should recognize its primary functions are to attract, motivate, facilitate, and retain talent. Teams should be given an identity, a vision, and quality goals.
- **Rule 6:** Teams should be organized with respect to member strengths and competencies.
- **Rule 7:** Dependable sources of knowledge should be provided in the form of textbooks and training materials.

Readers familiar with my May/June 2009 CrossTalk article will recognize two of these rules. Attention was also drawn to three essential ingredients of software development called the 3Ps: people, products, and processes [3].

Human Asset Evaluation

Not all assets are of equal value; so too can be said of human assets. The process of evaluation for most assets is based fundamentally on attributes and objective criteria, with a variable dash of subjectivity on top. Houses with identical specifications and locations (objective) can have different valuations due to the varying attraction of the décor (subjective). Likewise, diamonds of the same number of carats (objective) can have different valuations due to their luster (subjective). When assessing the value of human assets, we can follow a similar scheme.

The main objective criteria for evaluating professional (software) engineers are:

- Qualifications/education.
- Formal training.
- Experience.

If necessary, qualifiers can be used to turn these into quantitative criteria—consider the applicability, duration, and timeliness of each criterion. For example, a relevant degree is very valuable; if my experience is typical, this has shaped an engineer's aptitude from an early age.

Subjective criteria, sometimes referred to as *soft* skills, may include:

- Communication ability.
- Attitude/commitment.
- Adaptability.
- Assertiveness.

Decent soft skills are particularly relevant where team interaction and influence are important.

One may expect the formality of an employee evaluation scheme to vary with respect to the industry and the criticality of the application. With safety-related software systems in Europe, the range of individual and team competence is unacceptably wide. In recent years, I have had the pleasure of working alongside competent professionals with the requisite qualifications, training, and experience. I have also worked in critical environments where, clearly, the engineers lacked even basic skills. Some years ago, I was involved in an initiative to introduce a competence assessment scheme into a safety-related industry sector. A significant number of managers and engineers were unwilling to take part in the scheme until the objective criteria had been replaced by subjective criteria. Those same people would probably balk at the idea of being passengers on an airplane flown by a pilot who was not trained, but who had a pleasant voice over the speaker. Unfortunate as it seems, adopting formal employee evaluation or competence assessment schemes is like diet and exercise—we need it to stay fit and healthy, but doing the things that are best for us is not always easy. For safety-related systems, the British Health and Safety Executive has published guidelines for introducing a competence management system [4].

Could the certification of software engineering professionals be the answer (or part of the answer) in establishing competence? As is often the case when challenged to answer a software engineering question, we rely on respected sources of knowledge such as CrossTalk. Perhaps, surprisingly, there is a lack of information on professional certification; the obvious conclusion is that certification is at least an unpopular subject, or even taboo. Without a doubt, there are leading academics and practicing professionals who are concerned about the efficacy of certification programs. The core of the IEEE's Certified Software Development Associate and Professional efforts [5], the Software Engineering Body of Knowledge, has been criticized; for example, its ability to encompass all application domains is questionable. However, my own experiences (in various industry sectors in Europe) show a multitude of people, particularly in technical leadership roles, who would be fearful of successful certification initiatives that could expose their shallow grasp of a deep discipline.

Perhaps there are less formal and more palatable ways of assessing an engineer's competence. For instance, I have assembled a portfolio from the many projects I have worked on. This collection includes samples of requirements specifications, architectural designs, test specifications, process definitions, presentations, lists of technical books read and owned, etc. This gives me the ability to show someone the level of my experience and ability; yet, having a portfolio is far from common in our industries and it receives very mixed reactions. As DeMarco and Lister suggest, "It would be ludicrous to think of hiring a juggler without first seeing him perform" [2].

On the topic of hiring competent staff, professionals must be dismayed at the high proportion of job advertisements, over a substantial period, focused on low-caliber skills. For instance, experiences with a specific programming language or a particular requirements management or design tool are often cited as essential skills. With regard to requirements, the major skill is always in the specification: Tool proficiency can inject quality into requirements management, not the specification thereof. A skilled engineer will be trained to specify atomic, consistent, structured, and testable requirements; Wilson provides a synopsis on requirements specification in [6]. Similarly with design tools, and to quote Grady Booch, "CASE [Computer Assisted Software Engineering] tools have allowed merely bad designers to produce bad designs more quickly" [7]. The absolute fixation by whole industry sectors on programming language experience is a continuing embarrassment. There is an ever-present concern in our profession that the wrong categories of skills are encouraged and valued.

Judging from the feedback I received from my previous article, the most contentious areas were the rules and the section dealing with people—specifically, the contrasting behavior between professionals and amateurs. For a comparison, see Table 1.

We can take one of the divergences between these groups of people and analyze it further. Consider a sliding scale with art at one end and science at the other. If, by a process of task analysis, we conclude that the creative nature of software engineering and its resilience to practical mathematical proof places it nearer art than sci-

Table 1: *Characteristics of Software Professionals versus Amateurs*

The Professional Practitioner	The Amateur or Hobbyist
Views the overall task as an engineering discipline.	Describes the overall task as an art or craft.
Promotes a holistic, life-cycle view.	Holds an implementation, coding bias.
Places emphasis on the application or problem domain, and presents architectural solutions.	Places emphasis on the technical detail of the solution domain to the detriment of the customer or user.
Learns principally from published engineering literature.	Learns principally by emulating colleagues.
Encourages compliance with industry standards.	Prefers improvised, local procedures.
Employs quality criteria to manage projects.	Manages projects via schedule alone.
Conveys an outward, discipline focus.	Conveys an inward, project focus.
Exhibits a balanced approach to risk.	Adopts a naïve approach to risk.

ence, we must pause for thought. Good literature is founded on the discipline of strict linguistic standards (e.g., punctuation, spelling, and grammar), whereas music is founded on structures for tone, rhythm, and notation. History has proven, therefore, that discipline has released creativity, not stifled it: Discipline is as elemental to an artist, writer, or musician as it is to an engineer.

When a student receives a classical education in software engineering, this indoctrinates a view of, and an approach to, the discipline that is not just different from common practices, perceptions, and mythology—it is diametrically opposed. The gap between professional and amateur is not a gap, it is a chasm. Therefore, in comparative arithmetic terms, it is ludicrous to assume that 10 untrained personnel can perform even the work of a solitary professional. Perhaps this accounts for the 10 to 1 productivity ratio recorded as long ago as 1975 by Frederick P. Brooks in "The Mythical Man-Month" [8].

In 1980, I was a new graduate working in the British defense industry. I was approached by a concerned manager who observed, "You appear to be faltering and are not producing code as quickly as your peers." No one amongst this large office of software developers had ever witnessed a qualified and trained softie ratifying and specifying requirements, devising software architecture, designing the software, defining test cases and recording test results, as well as generating robust code. Wind the clock forward almost three decades and, in a high-dependability environment, our team was tasked to review multiple software requirements specifications produced by untrained personnel. Even with the availability of practical guidelines, the authors had produced worthless specifications. Combined with the worst that a bottom-up, functional software architecture has to offer, the project was in an undesirable state. The real problem in such a scenario is that, again, a group of untrained people cannot match the actual productivity of one professional. Equally, the majority will dominate proceedings and a solitary professional will toil to correct the substantial but substandard output of 10 untrained employees.

Lean Engineering is a populist topic, although it is well-documented that production-oriented techniques do not transfer readily into a (software) development environment—also known as the *make a cheeseburger, sell a cheeseburger* mentality discussed in [2]. Nonetheless, one of the principles of Lean is the reduction of waste in a production line. In terms of waste, having non-professional software personnel producing substandard artifacts is analogous to having an untrained team preparing and then selling raw, frozen burgers on a bun, with or without the cheese. In order to rectify this and similar waste issues, an organization may choose to assess the capabilities and training needs of project personnel—or choose to assess the competence of the management team that appointed and tasked untrained personnel in the first instance.

Software Defense Application

Readers will benefit from this defense software industry insider's focus on the proven, primary influence on software project success or failure: the combined capabilities of development team members. As for return on investment, enhancing team capability is unique in offering potential productivity gain factors of up to 10 times. Increasing team capability has the dual effect of improving the quality of software artifacts and reducing waste.

If the economic aspects of the software engineering life cycle were ever to be modeled, the most significant variables in the equation would reflect the human knowledge and experience. Then again, the life cycle has been modeled and the people capability attributes are the most significant. Barry Boehm identified this truism as early as the 1980s, and it is captured in COCOMO [9].

There are already significant clues here as to how to assess the value or competence of software engineering personnel. However, if your organization is still searching for a magical productivity enhancer, then look to laetrile[1] for the futility of searching for wonder cures, or "easy technological non-solutions" as described fully in chapter 6 of [2] and again in Brooks' "No Silver Bullet" chapter in [8].

Human Asset Enhancement

As we know, many assets (like cars) depreciate in value, while others (like real estate) increase in value—so we maintain, build on, and insure against loss those appreciating investments. Humans increase in worth through enhanced knowledge or experience and, likewise, an organization is prudent to invest time and money on its most precious people assets as well as to guard against the mishap of their loss. In our competitive technology-driven markets, companies striving to be the best have to attract, nurture, and retain high-caliber engineers.

I remember an organization that purchased each member of its administrative personnel a top-of-the-line PC and smart laser printer. What happened? The engineers stopped squabbling over their rented, one-between-five, basic machines—and looked on enviously. In this case, the organization showed a blatant disregard for its engineers. After this, many resumes were typed into those rented computers, including my own. When an organization equips all of its people adequately, provides an environment conducive to the intellectual task of technological development, and supports personnel growth, it in turn maintains, enhances, and protects human asset value. Engineers are people, and people need to feel their contributions are valued.

As indicated earlier, staff competence assessment schemes are often viewed negatively—but suppose such assessments were shown to link directly to an organization's development of, and investment in, its people. Therefore, the most effective teams can be organized on the objective basis of competence rather than on arbitrary and subjective opinions. As initial reluctance gives way to synergy, people will gel into strong teams.

Having assembled a strong team of competent professionals, how should it be organized? Brooks gives sound advice in his "The Surgical Team" [8] chapter. A surgeon and his surgical assistant perform an operation, while supported by nurses, an anesthesiologist, and administrative staff. This arrangement compares favorably with the roles of software architect, software manager, programmers, testers, and an administrator[2]. Consider the difference in the structure and formalism within different groups of musicians. A random collection of musicians can meet for a jam session; without proper direction or sheet music, the small group can still produce some decent, improvised sounds. Similarly, a small band of jazz musicians can, given reasonable levels of competence and practice, entertain an appreciative audience. However, to reach the excellence of a large professional orchestra is a monumental challenge in systematic coordination—the conductor on the rostrum and the sheet music are not just for show. For technical teams developing software-intensive systems of systems, the analogy is clear: Systematic and formal coordination is essential.

Having assembled a strong team of competent professionals, how should it be managed? Consider an environment where managers (verbally) whip teams to meet implausible milestones; where engineers are forced to cut corners and undermine basic quality criteria; where training is regarded as an unnecessary expense, and reading a book is considered a waste of company time; where professional opinions are most unwelcome and people are expected to just get on with the job; where political mendacity is a substitute for competence; where knowledge is wielded like a bludgeon with people herded into pens like cattle; in an office where it is too hot, cold, or noisy for anyone to function efficiently. In contrast, imagine an environment where people are encouraged to design outstanding products; where teams and individuals are challenged to excel; an office with shelves groaning from the weight of books by software gurus; where training is viewed as a necessity; where professionalism is a given; where diverse opinions are seen as the balance in an open political culture; where personnel are provided with adequate tools and comfort; and a place where the organization and staff have mutual goals and aspirations. In summary, it is as easy to obtain the least value from your human assets as it is to obtain the most value.

Conclusion

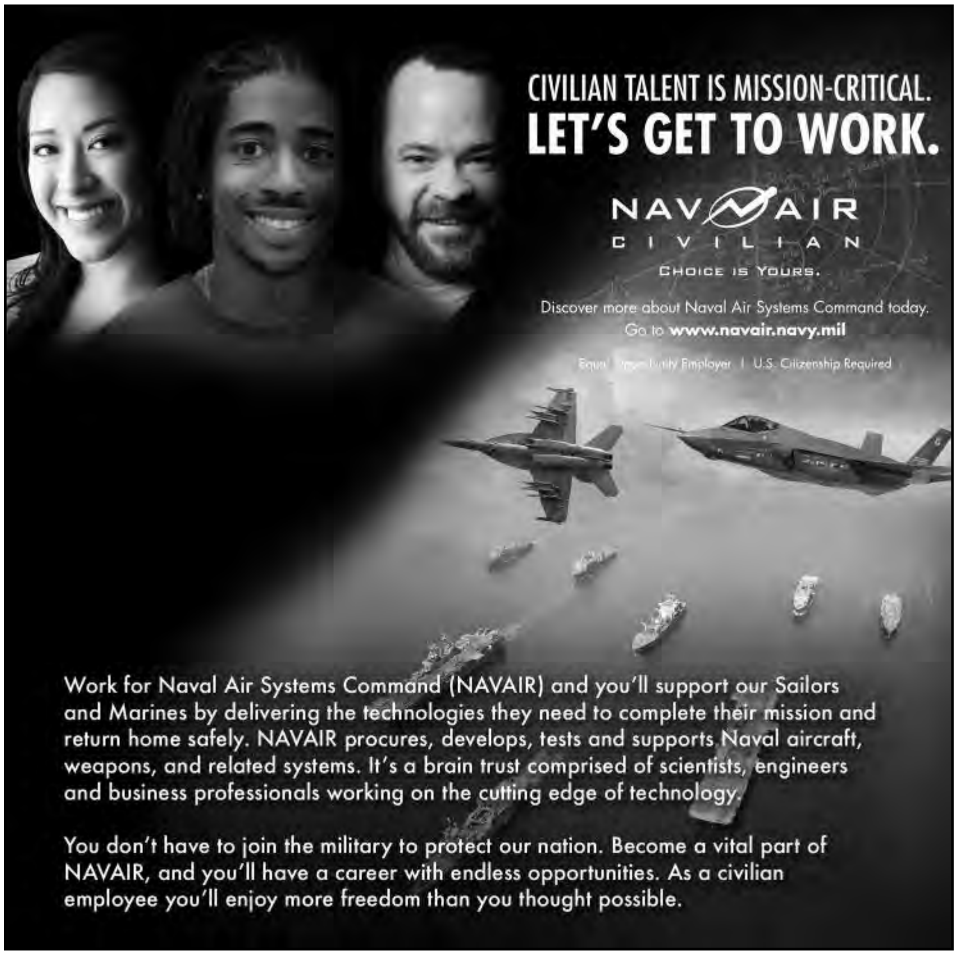
The legacies of Adam Smith and Robert Owen are an important reminder to us that people are at the heart of commercial and social success. In our rapidly changing technological world, it is worth considering their centuries-old wisdom. Perhaps there is an opportunity for our organizations to look again at the value, rather than the cost, of their people assets. When people are viewed truly as vital assets, then investment in them is sure to deliver a mutually beneficial corporate future. This will, in turn, lead to greater customer satisfaction.◆

References

1. "New Lanark – World Heritage Site." <www.newlanark.org>.
2. DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams.* New York: Dorset House, 1999.
3. Allen, Martin. "From Substandard to Successful Software." CrossTalk May/June 2009 <www.stsc.hill.af.mil/crosstalk/2009/05/0905Allen.pdf>.
4. Health and Safety Executive, The Institution of Electrical Engineers, and The British Computer Society. *Managing Competence for Safety-Related Systems.* 2007 <www.hse.gov.uk/humanfactors/topics/mancomppt1.pdf>.
5. IEEE Computer Society. "Certification and Training for Software Professionals." 2009 <www.computer.org/portal/web/certification>.
6. Wilson, William M. "Writing Effective Natural Language Requirements Specifications." CrossTalk Feb. 1999 <www.stsc.hill.af.mil/crosstalk/1999/02/wilson.pdf>.
7. Booch, Grady. *Object-Oriented Analysis and Design with Applications.* Menlo Park, CA: Addison Wesley, 1994.
8. Brooks, Frederick P. *The Mythical Man-Month: Essays on Software Engineering.* 20th Anniversary Edition. Reading, MA: Addison-Wesley, 1995.
9. Boehm, Barry W. *Software Engineering Economics.* Upper Saddle River, NJ: Prentice Hall PTR, 1981.

Notes

1. Laetrile is an extract from apricot stones, sold in Mexico as a (fraudulent) cure for cancer.
2. The role of software manager here is one of a facilitator, rather than a technical lead. Team composition should also vary with the magnitude and criticality of the task.


CIVILIAN TALENT IS MISSION-CRITICAL.
LET'S GET TO WORK.
NAVAIR CIVILIAN
CHOICE IS YOURS.
Discover more about Naval Air Systems Command today.
Go to www.navair.navy.mil
Equal Opportunity Employer | U.S. Citizenship Required
Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.
You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.

About the Author

Martin Allen is a software engineering professional with 30 years experience, mostly in the defense industry in the United Kingdom. He has worked on many successful software-intensive systems for the British Royal Air Force and the Royal Navy. Allen has always had a strong interest in industry standards for the engineering of dependable systems. His other professional interests include risk management, software cost economics, requirements analysis, design methods, and software testing. Allen and his colleagues work on the boundary between the academic research of computer science and the practical application of software engineering.

1 Belfry Walk
Titchfield Common
Hampshire, United Kingdom
PO14 4QD
E-mail: mjallen60@yahoo.co.uk