21st Century Engineer

Lynn Robert Carter
Software Engineering Institute
Lt. Col. Scott Dufaud
Air Force Y2K Program Office

Editor's note: The following is an excerpt from a forthcoming book by Lynn Robert Carter of the Software Engineering Institute and Lt.Col. Scott Dufaud.

Foundations

It is difficult to find any single definition of engineering that generates uniform agreement, but many are similar to the following:

"*The art of the practical application of scientific and empirical knowledge to the design and production or accomplishment of various sorts of constructive projects, machines, and materials of use or value to man* [1]."

For most of history, this definition would not have been useful. It was not until the middle of the 18th century that the first civilian engineering school was created [1], and that education program was little more than a structured form of apprenticeship with very little dependence on mathematics, physics, or other related scientific fields. From this beginning to the middle of the 19th century, the addition of the scientific method, the creation of professional societies, and the growth of engineering schools led to the golden age of engineering — the middle of the 19th century to the middle of the 20th century [2]. This is not to say that many fine examples of engineering had not been created before then, but the labor was more craft than profession. Early work was the culmination of solid common sense and the wisdom of experience, while the golden age benefited from the application of scientific principles. It is also important to recall that a handful of traditional engineers would produce the drawings and guidance to potentially hundreds or thousands of skilled craftsmen on the factory floor. It was there that physical reality was created and it would be foolish to ignore the contributions of these laborers beyond just performing the work described by the engineering department.

The middle of the 20th century brought about many changes, including a rapidly growing technology base, an equally growing consumer marketplace, and the advent of computing. With this explosive growth came a shift in public opinion about engineering. The creators of the many marvels that had transformed our lives were no longer perceived as the source of solutions to society's problems [2]. While it is unfair to unload all of the blame on computing, the issues of complexity and speed of change — hallmarks of computing — along with a naïve understanding of engineering are the likely root causes. Engineering grew from a skilled craft based on apprenticeship to a profession based on rigorous engineering school education and careful development programs in a myriad of engineering departments. Little of either of these remain on the path many take in today's world of more, cheaper, faster.

The change did not happen overnight. In the early days of computing, the shift from a classical engineering discipline began. To the United States military, the computer provided solutions for a world full of lethal enemies and increasing nuclear powers. From the calculation of trajectories for big guns, to the fusion of data to detect first-strike missile launches, no mechanical or single-function machine could fit the bill. A calculating machine with modifiable functionality was precisely what was needed and the United States military took the lead. From the 1940s through the '60s, the state-of-the-art in computing was defined by the work produced by or on contract for the military. While several noted institutions of higher education provided specific technical courses in computing, the bulk of the software was developed by engineers from the classical engineering fields or by mathematicians. As the military continued to find ever more sophisticated ways to employ computing, a trend became obvious. A large portion of the total life cycle costs of systems shifted from hardware to software at an alarming rate, far faster than new software developers were being produced.

As the complexity and costs of systems shifted from hardware to software, the need disappeared for a factory of skilled craftsmen to transform the output of engineering into physical reality. A hidden downside was the loss of a critical system review by involved but independent people. It was common for factory personnel to uncover problems and resolve them before the customer saw the result. The lack of effective feedback mechanisms to engineering was one of W. Edwards Deming's complaints about North American manufacturers [3] and now the people best positioned to provide that feedback were being eliminated. Where the skillful eye and ingenuity of the factory worker had served to back up creative engineering departments, new products are assembled by compilers, linkers, and computer-driven duplicators, none of which have common sense or insight about the real need of the customer. It is true that this automation can be less costly and far faster; it assumes the output of engineering is perfect. If it is not, a critical quality control function was eliminated and few firms took the initiative to effectively place those old responsibilities onto engineering.

Prior to the shift toward software, quality was built into the product through a self-imposed process where engineering on paper and tube-bending realism had to meet and work. In addition, turning out more product was largely a manpower issue. Leaning on the manufacturing process to turn out more product only required working longer hours, working more people (maybe in shifts), or both. If the factory employed skilled workers, provided them adequate tools, and allowed them to build quality products, quality products resulted, even if the drawings from engineering were flawed. Software does not fit this paradigm. Because software quality is not a characteristic of the physical world, this aspect of the product could not be determined by visual inspection by either the producer or the consumer. Quality was no longer a partnership of the engineering department and the factory floor with its skilled craftsmen. Quality is an exercise in mental understanding of what the product is supposed to do and shaping creative possibilities into code. In the software paradigm, turning out more product also is different as the market is always looking for the next release with a set of new features. No longer does using more factory workers putting in more hours turn out more quality product. In order to do this, we must now lean on engineering. In response to the need for more software, many elected to hire more developers and compressed the time line to develop them. This has led to an even further degradation in the spectrum of quality, with fewer and fewer software developers receiving even the most basic formal training or skill development in the application of scientific methods to their work. Without the formalisms or the apprenticeships of old, where are they supposed to develop their skills?

Situational Assessment

We have become so used to poorly designed software and throw-away hardware that we do not even think about it. While it is common for the military to think in terms of 30- to 50-year weapon system life spans, the commercial off-the-shelf (COTS) hardware and software systems used to feed data to these weapon systems and to link them together via networks are nearly obsolete the day they are installed. In addition, the lack of long-term vendor support seldom seems to be a concern. Sadly, quality problems are not just problems in the COTS hardware and software. The destruction of the space shuttle Challenger (faulty seal design), the distortion in the Hubble images (improperly ground lens), the slowly degrading capability of the Patriot missiles during the Gulf War (software error), and the long string of Delta launch vehicle failures (cause not publicly known) are all high-profile examples of quality problems from firms from which we expected more. Rather than designing-in quality up front, the attitude of many seems to be to test quality in with huge beta-test efforts, even in the face of mathematical proof that such testing can not ensure quality.

Even more disturbing is the growing dependence of the military on technologies, such as those supporting the World Wide Web and the Windows operating system, that have been repeatedly shown to be vulnerable to even teenage hackers. What usually goes unspoken is what could be accomplished by an elite information warfare team. Since these technologies were designed for commercial applications by commercial firms addressing a commercial marketplace, many issues critical to our war-fighting military were not design requirements. As a result, our military services struggle to obtain and keep enough qualified and skilled network engineers, to keep these critical systems running, and to keep them secure. Some of these people can cost more per year than the cost of the computing systems on the network they support. Better network implementations are being pursued, but in the current climate of "more with less," these are being addressed in a piecemeal fashion, at best. Attributes like quality, standardization, and repeatability are most often found in organizations with people having strong personalities that believe in these attributes. That is to say, these organizations are personality-driven and these attributes are part of those personalities as opposed to being core attributes of the overall organization. The problem with this situation is that when the proponent leaves the organization or is forced to reorder his or her priorities, the organization's products and processes cease to possess these characteristics.

This situation results in wasted resources and increased mission risk when we can least afford it. The shift of costs from hardware to software has led to a shortage of adequate software development capacity, labeled "the software crisis," and it continues to grow at rates commensurate with the increasing complexity of our systems and technologies. While many have fought to bring the title "engineer" to the field of software development as the solution to this crisis, the pressures of the marketplace and rapid rates of technology change have resulted in an interesting split. We use the phrases "blue-collar software developer" and "gold-collar software developer" to highlight this split.

> The software crisis continues to expand at rates commensurate with the increasing complexity of our systems and technology.

Two Paths to the Future

The trends seem clear. Consumer demand for ever more sophisticated systems at lower costs will continue to drive commercial and military development. In turn, the development community must take a similar economic approach for solvency. Survival will depend upon the ability to increase efficiency, shorten cycle time, and focus on the bottom line, without sacrificing quality. As we have indicated above, there appear to be two very different approaches to address this trend: blue-collar and gold-collar development strategies. We offer the following distinctions to differentiate the competing philosophies.

The first difference between these developers can be found in how their employers go about addressing the need for more output. Blue-collar organizations believe the solution is to hire more workers and find ways to keep the cost of each as low as possible. The hiring process is little more than verifying that the developer has the needed skills, is

The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark office to Carnegie Mellon University.

willing to work long hours, and is willing to take lower pay in exchange for stock. (It is sad to note how few engineers are able to redeem their stock for cash and recover the income they have deferred.) Gold-collar firms look for ways to better leverage their existing resources by eliminating waste, obtaining better tools, and ensuring that barriers to progress have been removed. Fast, flexible, and cost-effective solutions seldom occur on projects buried under developers and gold-collar firms always look for alternatives to the "piling-on" approach to shorten the development time. Watts Humphrey has stated that performance differences of factors greater than 10 have been seen between organizations with basically the same business, workers, and tools. This observation led to his effort to better understand these differences and to develop methods to leverage the understanding. From this came the Capability Maturity Model® (CMM®). Again and again, the Air Force and commercial firms have shown that the application of these concepts can produce dramatic results. Yet many firms still look for large numbers of people with the right design, coding, and testing skills while they ignore the nonsoftware skills required to progress up the CMM levels.

The second difference is the attitude toward process by the people in a firm. Blue-collar development firms talk about the costs and the delays associated with the overhead of process and the damaging constraints it places on creativity. Gold-collar development firms point out the dramatic improvements in productivity and quality found in a balanced focus, including process. Having reaped the benefits, they are somewhat amused by the notions that process slows development or constrains creativity. Firms that have successfully climbed CMM levels report amazing returns on their process improvement investments and point out how process reduces waste, allowing their developers to spend more time developing code likely to appear in finished systems. The results from the shuttle on-board software engineers align nicely with the results from Boeing and numerous others. Yet, blue-collar firms remain unswayed by these results.

The relationship between a firm and its customers is the third difference between these two philosophies. Blue-collar developers are kept far away from the ultimate users with buffers of managers, marketers, salespeople, and support personnel. There is a concern about loss of control and the need to keep the blue-collar developer focused on writing code. Firms obtaining full benefit from gold-collar developers see the need to have their developers fully integrated with the customer. From these firms' perspective, technology has decreased cycle time, and any nonvalue-added role in the development process can only slow things down. As technology grows in complexity, the challenge of working closely with customers grows in its importance. Blue-collar firms point to the lack of appreciation many customers have about current technology issues, the lack of clarity about product requirements, and their unreal expectations about the time and effort to accomplish the ill-defined work. These, they say, are reasons enough to maintain this distance. Gold-collar firms point to the same things and assert that this demonstrates the need to be closer. Some go even further, insisting that they must help develop their customers to ensure higher mutual satisfaction in the future. They advocate the use of the spiral model in development and in customer relationships, and trust in their gold-collar workers' use of process to keep things under control. Blue-collar firms scoff at such notions as wishful thinking and a waste of time and resources in a turbulent world. After all, they claim, we are talking about human beings, human nature, and no one can make guarantees about which firms will exist in the future, let alone who will be the market leader.

The use of scientific methods, mathematics, risk management, and lessons learned are the fourth major difference. Gold-collar workers easily separate work into well-known and novel classes of effort. Effort and work can be classified as either, "we have already done this," or "this is something new." With a solid discipline for employing process and data, they strive to leverage the work of others as well as their own experiences to build an ever-growing foundation. When results differ from expectations, they take time to understand the root cause and employ mechanisms to reduce the probability of similar difficulties in the future. Their process focus provides the basis for using statistical methods. This supports understanding common vs. special causes, allowing effectively designed improvements to address the former. Blue-collar firms point to the decrease in cycle time and the increase of complexity as proof that nothing stays the same long enough for statistical methods to be useful, even if there was enough extra resources around to waste on process and data capture.

The fifth major difference is in the perception of the skills workers need to have and the best way to organize and manage those workers. Blue-collar organizations tend toward numerous groups of specialists and organizational structures to ensure the work gets done. There are clear lines of authority and control in order to ensure no one is working on things other than what is currently required. (In many such organizations, few of those in a position of authority and control are particularly good at this. Few blue-collar firm managers are able to produce a list of projects under their control or indicate who is working on what.) Gold-collar organizations strive to populate their work force with generalists, each having one or more areas of unique skill that complements others. The goal is to have small, customer-involved, self-directed work teams. They leverage modern technologies and tools, but most importantly they leverage the other team members to do it all. They think that such tight integration of small teams leads to faster results with products that better fit the customers' needs, including making it harder for operational mistakes to occur because the common cause for such errors have been designed out of the system. (After all, gold-collar firms know how to use statistics to their benefit as well as to their customers' benefit.)

Choices to be Made

Looking at the exponential growth curves, it is clear that something will have to change. Gordon Moore, of Intel, observed that the density of storage, in bits per square inch, from integrated circuits appears to be doubling each year to 18 months. This observation has been called Moore's Law [4], and it appears to apply, in modified form, to the doubling of the number of engineers it takes to produce each new generation of Intel processors. In not too many more generations, if these trends continue, everyone in the United States will be a processor designer for Intel. This should be a clear signal that the approach of many blue-collar firms will not work. Even if they could find people to hire, many are reaching the point of diminishing returns. The easiest approach of hiring more people is not really a reasonable long-term solution. A second approach is to acquire and employ higher performance tools. At first glance, this seems more reasonable. The capability of modern software tools can be very empowering. The question that must be asked revolves around the long-term benefit of these tools. Tektronix was one of the first firms to employ high-level languages in the development of microprocessor software in embedded systems using a retargetable compiler system. The Tesla programming language was used in more than 60 products over the years, and its gamble paid off. The decision to leave Tesla for C was a difficult one, but the lack of a large enough pool of developers willing to learn Tesla forced the change. No technology will remain in first place forever. The trick is to pick technologies likely to remain useful long enough for the firm to recover its investment as well as to place itself in a position to leverage the next big technological advancement. This can be very difficult. (Ask the people who selected Beta over VHS because of the superior picture Beta delivered.)

Another aspect of a tool-based approach to the problem lies in the nature of high-performance tools. Long-term benefit from these tools requires discipline throughout the development process. Most of these tools require a heavy investment during the early phases of the life of the project in order to yield big wins at the end. While blue-collar workers may have the skills to use these tools, their firms often lack the wisdom to stay the course charted at the beginning of the project. Decisions are made with no appreciation of the impact on the project. Gold-collar firms, with their disciplined processes, are able to appreciate the impact at the end of a project to a decision being made much earlier. This does not imply they will not make bad decisions, but at least the issues surfaced and an informed choice was made. Short-term decision making is more often based on considerations that have little to do with the projects that are ultimately affected by those decisions. Political, economical, and personality-based short-term decision making is a symptom of a blue-collar mentality that says we can overcome any setback with more people and more hours. The gold-collar mentality understands that good project engineering is too important to the enterprise and must be bullet-proof to these other considerations. As a result, gold-collar practitioners have processes and set ground rules that minimize the chance their projects will be derailed by considerations outside of the team's control. Too often, blue-collar firms are surprised by the failure of their new tool, cannot see what caused the failure, and reach the faulty conclusion that the tool does not work. Ineffective implementations are at the heart of a majority of tool adoption and process improvement failures [5].

Large-scale changes in the way we do business are often thwarted at the highest levels of leadership because the new paradigms are not compatible with the thinking and ideas that put our leaders where they are today.

From our position, the optimal approach is to chart a course toward becoming a gold-collar development firm leveraging gold-collar developers utilizing gold-collar processes. While we acknowledge the accomplishments of heroics, we see the long-term costs. Success comes from a wise combination of tactical skill and strategic wisdom. Championship pool is an excellent metaphor. It is not enough to have the skill needed just to make the next shot. True champions know where they want to leave the cue ball for the next shot as well as the rest of the shots on the table. Gold-collar workers are able to see the consequences of their actions and regularly leave themselves and their project teams better positioned for the next project than they were for the last. At the heart of a gold-collar firm is a focus on effectively leveraging the lessons others have learned. We believe an effective deployment of process is the method of choice to accomplish this.

As individual contributors, the choice should also be clear. The days of the solo developer coding a sequence of market-winning products are over. (That is assuming they ever existed.) The future is in high-performance teams of flexible gold-collar workers striving to bring real engineering to software development. As quoted above, a critical aspect of engineering is the "... application of scientific and empirical knowledge...." Being an engineer is not about what you know. It is how you go about applying what you and others know and what you can learn.

A 21st century engineer is more than someone with a set of skills, including process skills. These skills must be balanced with discipline and wisdom needed to be an effective member of a high-performance team. The Navy's elite flight demonstration team, the Blue Angels, is an excellent metaphor. A high-performance team does not happen overnight simply because a collection of experts is told it is a team. Building a team requires the right kinds of individuals and the right assembly process. Once again, poor implementations have given team-building exercises a bad name, but one can learn a great deal from the Blue Angels. Year after year, 50 percent of the team rotates and year after year the new team is formed. A critical aspect of the team's success is the time it spends becoming a team before its first public performance of the year [6]. If carefully selected team candidates are guided through truly meaningful team-building exercises, it is reasonable to assume the results might be as stunning as those of the Blue Angels.

Implementation Success

The concept of high-performance teams is not new or unproven. There are numerous examples from recent United States history where highly disciplined teams consisting of thousands of engineers not only undertook what were considered impossible goals, but succeeded. The Manhattan Project produced the nuclear bomb. The teams at NASA not only produced the Apollo series of space missions that put a man on the moon in 10 years, but also produced the incomparable Shuttle program. An oft-cited group, Lockheed's Skunkworks Division, produced the SR-71 Blackbird, a feat of technology that was three generations ahead of its day. In many ways, most of the military technology today is a result of high-performance teams leveraging state-of-the-art technology in ways previously unimagined. We believe it is our next task to make the creation of these teams and their successes more commonplace and at the basic level of all systems engineering, not just for high-value military systems.

There has been progress in software. Many technology consulting firms also possess high-performance teams aimed at improving the state of systems or enterprise engineering. Andersen, Deloitte & Touche, and Coopers & Lybrand, as well as other look-a-like companies have created their own brand of enterprise-engineering services. The Air Force also has had its share of success developing organic high-performance gold-collar teams. The point is that these teams already exist, as well as the process for creating them. Three examples help demonstrate our assertions.

Software Process Improvement

In the early 1990s, when the need for military software and software resources underwent a huge expansion, the Air Force made a deliberate decision to pursue excellence in the way it developed and maintained software. What followed was almost a decade of cooperation with the Software Engineering Institute in order to improve the state of Air Force software development. The CMM for software essentially was institutionalized across all Air Force software houses as well as within the contractor communities supporting the Air Force. In 1999, even though philosophical changes forced most of the Air Force away from primarily organic software efforts, organizations still controlling their own software projects retain a strong recognition and desire to use mature development techniques and processes as defined by the CMM. Such institutionalization is a result of the early commitment the Air Force made to create lasting results via a high-performing Software Process Improvement (SPI) team. There was a commitment to provide the people and resources necessary to develop critical skills and disciplines, build the programs necessary to take the fight to the units in the field, fund the temporary duties needed to work face-to-face with commanders, and stay the course in the face of early efforts that appeared, on the surface, to be failures. As results showed progress, the SPI team regularly evaluated options for improvement, with new capabilities and services continually added to their toolbox. The result has been impressive. Even though the SPI team officially disbanded in 1997, the cultural change within the Air Force software community has been surprising. At the height of the SPI team's activities, the most common remark heard from commanders was that there was not enough time and resources to make SPI happen via CMM. Almost three years later, we continue to hear about current commanders — those who were O-1 to O-4 at the height of the SPI effort — who claim they cannot develop good software without the guiding principles provided by the CMM.

The future is in high-performance teams of flexible gold-collar workers striving to bring real engineering to software development.

Year 2000

When the Year 2000 (Y2K) problem emerged as a serious threat to Air Force systems, it became a top priority almost immediately within the Communications and Information communities. Again, the Air Force made a conscious decision that Y2K must be urgently addressed by the highest levels of leadership within the Air Force. The Air Force Y2K Program Management Office (PMO) was developed and patterned after the Air Force SPI team. (Some personnel were on both teams.) The Air Force Y2K effort would utilize the guiding principle of centralized Air Force management and decentralized execution at the field and organization levels. Therefore, the processes and tools the PMO developed were key, as they would serve as the standard baseline for all Y2K remediation efforts undertaken by units in the field at all the various levels within the hierarchy. It is interesting to note that as in the SPI example, the Air Force was out in front of the other Department of Defense components for the Y2K effort. Many of the products, processes, training, and tools created by the Air Force PMO were subsequently adopted and used by various other governmental organizations for use in their respective Y2K efforts. It is noteworthy that many of the Air Force PMO-developed processes and tools were based on the concepts and practices defined by CMM. As in SPI, the Y2K PMO team's primary role was to develop the best practices and then shepherd the rest of the Air Force in using practices through face-to-face interaction that would include training, consulting, inspecting, and feedback conferences.

Network Management

The emerging network installation and network management teams provide a final Air Force example. At a time when networks are proving to be a most valuable and critical resource, network performance is sporadic and less than optimal. The Air Force response once again has been to turn to a high-performance gold-collar team to ensure that all base networks work, and work well. The highly trained SCOPE Network teams specialize in network management disciplines and have been created to standardize and optimize the performance of all Air Force base networks. These gold-collar teams travel to all sites to perform fine-tuning of networks, ensure adequate security, improve operations, train local personnel, and share best practices. As in the previous two examples, they utilize a disciplined process as they perform their work. As the Air Force experts, they shepherd the training and development of all network management personnel through face-to-face interaction, which includes training, consulting, and feedback. The SCOPE Network effort has proven to be a highly effective use of gold-collar development teams and has the data to prove its success.

If we step back and examine these three examples with respect to the differences described earlier, we find some interesting commonalities. Each of the examples required:

- dramatic improvement in specialized functions
- changes of core attitudes toward the use of disciplined processes (in the first two, this required changes of priorities by command-level leadership)
- merging of disparate, and sometimes confrontational, consumer and producer groups into integrated interdisciplinary teams with common goals
- use of the latest scientific methodologies, data and statistics, risk management, best practices, lessons learned, and other applicable leading-edge technologies into standardized, predictable, and repeatable processes
- change of old patterns of thought that people can only employ one or two specialized skills

Conclusion

From experience, it appears that the successful application of high-performance gold-collar teams has less to do with the technology being implemented than it does with other factors. Deciding if and when a gold-collar team is required is a crucial task that should not be taken for granted and is at least as important as the implementation of such teams.

We offer the following considerations as key criteria for helping to determine when high-performance gold-collar teams are an appropriate enterprise strategy:

- when an enterprise has deemed a capability or asset to be at the core of its capability to compete and survive in the marketplace;
- when an enterprise action is a priority that must meet specific performance levels, meet schedule time lines, and/or meet cost constraints.

The key ingredient, crucial for success, is twofold. First, there must be an identifiable need that can be articulated to the senior decision-makers of the enterprise in a way that causes them to take some action outside of the status quo. Second, there must be a conscious decision that the enterprise will pursue the gold-collar strategy for the long run. Once the decision is made to pursue, the commitment to its end must be total. These strategies are no longer off-the-wall experiments to achieve extraordinary results. If the enterprise will make the long-term investment, the results will be there in the end.

In all of the examples cited here, each has a common, yet subtle, characteristic that cannot be overlooked. Each of the situations required a change in the mindset of the prevailing culture of the day. All of today's leaders grew up with a frame of reference that was generally two to three iterations or more behind today's. As a result, large-scale changes in the way we do business are often thwarted at the highest levels of leadership because the new paradigms are not compatible with the thinking and ideas that put our leaders where they are today. This single fact is a primary reason why the success of efforts like these are so dependent upon the ability to interact face-to-face and work collaboratively, in a process-disciplined manner, to implement the change. Human interaction and the ability to manage resistance to the effort is as key to the high-performance team as any other aspect.

Final Thought

If software is mission-critical, design an environment to ensure it works. The decision to outsource is easy, but the cost to outsource and to ensure mission-critical capabilities survive is not trivial. Firms that outsource should follow the Air Force's approach to outsourcing the development and manufacture of fighters and bombers. The best pilots the Air Force has are intimately involved in every step of the process. What is the equivalent in your organization for your software-intensive systems? Gold-collar firms know when it is appropriate to outsource and they bring the same discipline to that task as they do to all the others. It is time for us all to make some critical choices and make the future a reality.

About the Authors

Lynn Robert Carter is a senior member of the technical staff of the Software Engineering Institute. From his office in Phoenix, Ariz., he supports the adoption of technology at a number of customer sites as a vehicle to learn how technology adoption can be made more predictable and less costly. Carter's focus is currently on the roles and responsibilities project leaders, managers, and executives must honor in order to establish environments conducive to the adoption of new technology without sacrificing mission capability. Carter has served as the director of Systems Engineering at EdgCore Technology, as president and CEO of Network Solutions, as a researcher at Motorola, and as a principal engineer at Tektronix. Carter served as an officer of the IFIP Working Group (2.4) on System Implementation Languages for 14 years, has published numerous papers, and has written and co-written three books. He received his bachelor's and master's degrees in mathematics from Portland State University and holds a doctorate in computer science from the University of Colorado at Boulder.

Software Engineering Institute
Carnegie Mellon University
3857 East Equestrian Trail
Phoenix, Ariz. 85044-3008

Voice: 480-598-1247
Fax: 480-496-9464
E-mail: lrc@sei.cmu.edu

Lt. Col. Scott Dufaud is deputy program manager for the U.S. Air Force Year 2000 Program Management Office at the Air Force Communications Agency (AFCA), Scott AFB, Ill. Prior to assuming these duties in November 1996, he was the chief of the Software Management Division at AFCA. Dufaud specializes in software management issues, software engineering process groups, software process improvement via the Capability Maturity Model, technology insertion, and issues of accelerating organizational change. He previously served at Headquarters Strategic Air Command and U.S. Strategic Command, the Air Force Manpower and Personnel Center, and Headquarters Air Force Space Command. He has a bachelor's degree in computer science from Southwest Texas State University and a master's degree in systems management from the University of Southern California.

HQ AFCA/ITY
203 W. Losey St, Rm 1065
Scott AFB, Ill. 62225-5224
Voice: 618-256-5697 DSN 576-5697
Fax: 618-256-2874 DSN 576-2874
E-mail: scott.dufaud@scott.af.mil

References

1. Kirby, Richard Shelton, et. al., *Engineering in History*, Dover Publications, New York, 1990.
2. Florman, Samuel C, *The Existential Pleasures of Engineering*, Second Edition, St. Martin's Griffin, New York, 1996.
3. Deming, W. Edwards, "Out of the Crisis," M.I.T. Center for Advanced Engineering Study, Cambridge, Mass. 1991.
4. Raymond, Erics S., editor, *The New Hacker's Dictionary*, The MIT Press, Cambridge, Mass. 1991.
5. Rummler, Geary A., Alan P. Brache, *Improving Performance*, Jossey-Bass Publishers, San Francisco, Calif., 1995.
6. Rear Adm. Moneymaker, Patrick D., personal conversations with a former commander of the Blue Angels.

Coming Events

16th International Conference on Data Engineering
Dates: Feb. 28-March 3, 2000
Location: San Diego, Calif.
Sponsor: IEEE Computer Society
Web site: http://www.research.microsoft.com/icde2000/

13th Conference on Software Engineering Education and Training
Theme: Software Engineering Coming of Age
Dates: March 6-8, 2000
Location: Austin, Texas
Sponsor: IEEE Computer Society
Web site: http://www.se.cs.ttu.edu/CSEET2000/confhome.htm

12th Software Engineering Process Group Conference (SEPG 2K)
Theme: 2000 Ways to Make Software Better
Dates: March 20-23, 2000
Location: Seattle, Wash.
Sponsor: Software Engineering Institute
Web site: http://www.sei.cmu.edu/products/events/sepg/

FOSE Leading-Edge Technology for Leaders in Government
Dates: Apr. 18-20, 2000
Location: Washington D.C.
Topic: FOSE offers a variety of educational opportunities for all levels of IT professionals.
Web site: http://www.fedimaging.com/conferences/

12th Annual Software Technology Conference
Theme: Software and Systems — Managing Risk, Complexity, Compatibility, and Change
Dates: Apr. 30-May 4, 2000
Location: Salt Lake City, Utah
Co-Sponsor: U.S Air Force, U.S. Army, U.S. Navy, U.S. Marine Corps, Defense Information Systems Agency, Utah State University Extension
Contact: Dana Dovenbarger
Voice: 801-777-7411
Fax: 801-775-4932
E-mail: dana.dovenbarger@hill.af.mil

First Software Product Line Conference (SPLC1)
Dates: Aug. 28-31, 2000
Location: Denver, Colo.
Sponsor: Software Engineering Institute
Web site: http://www.sei.cmu.edu/plp/conf/SPLC.html

23rd International Conference on Software Engineering
Dates: May 12-19, 2001
Location: Toronto, Ontario, Canada
Sponsor: IEEE Computer Society TCSE and Association for Computing Machinery SIGSOFT
Web site: http://www.csr.uvic.ca/icse2001/