Commandments for a Productive Development Environment

Dr. Randall Jensen and Les Dupaix
Software Technology Support Center

The software development industry has spent decades – with little success – attempting to make large productivity improvements through technology changes. But some projects have broken the productivity barrier by applying common sense practices to the people side of the development process. This article gives a set of commandments from lessons learned from projects with major productivity successes.

For more than 40 years, the software development industry has tried to improve productivity by implementing technology advances like the following:

- Third and fourth generation programming languages.
- Structured techniques (functional and object-oriented).
- Process variations (waterfall, rapid application development, rapid prototyping).
- Environments (programmer's workbench, .NET).
- End-user programming.

Some technologies have worked well. For example, the introduction of higher order languages (FORTRAN, etc.) reduced the size of software programs by as much as 70 percent. Despite this gain, however, if we measure the cost of developing a single source line of code from development start through product sell-off, we find that over the last 40 years productivity has increased an average of only one source line of code per person-month per year. That is, the average productivity for Department of Defense software has only improved from about 60 lines per person-month in 1960 to about 100 lines per person-month in 2000 for similar products. Thus, we see technology advances, including structured techniques, Computer Aided Software Engineering (CASE) tools, modern development environments, and process maturity have not provided the gains we anticipated.

Figure 1 [1] illustrates the vigor with which we have pursued a technology solution (silver bullet) to the productivity problem. The key to increased productivity must therefore be elsewhere. Weinberg demonstrates this by comparing the relative percentages of Software Engineering Institute publications in major activity areas of technology (tools), people (education), systems (development environments), and management to the relative productivity gain for each group. According to Weinberg, the most significant productivity improvement area is, by far, the manager activity area.

Barry Boehm argues, "Poor management can increase software costs more rapidly than any other factor." But he explains in the following:

> Despite this variation, COCOMO [constructive cost model] does not include a factor for management quality, but instead provides estimates which assume that the project *will be well managed.* [2, italics per the article authors]

Well managed does not work in this context. Without the management factors, we cannot distinguish between well-managed and poorly managed projects. Looking at the results from the 2004 Standish Chaos Report [3], most projects are not well managed today. The report divides projects into three classes: successful, challenged, and failed. About 28 percent of the projects evaluated were classified as successful, albeit they delivered an average of only 52 percent of the original requirements. Fifty one percent were delivered, but with significant overrun in cost and schedule while delivering only a fraction of the original requirements (challenged). About 18 percent were cancelled before delivery (failed). In other words, ignoring management factors in an estimating tool means that the projects are *consistently* not well managed. All projects have problems, but most often they are people problems rather than technological problems.

Recognizing the importance of good management in software development productivity is only the first step in process improvement. Moreover, good management is more than management style and organizational ability. Good management requires effective communication. Effective communication is, thus, essential to successful software development productivity gains.

This article will discuss team communication and management issues within the development environment and their effect on software productivity. Solutions to communication problems are largely common sense, can be implemented with minimal investment, and have almost immediate payoffs.

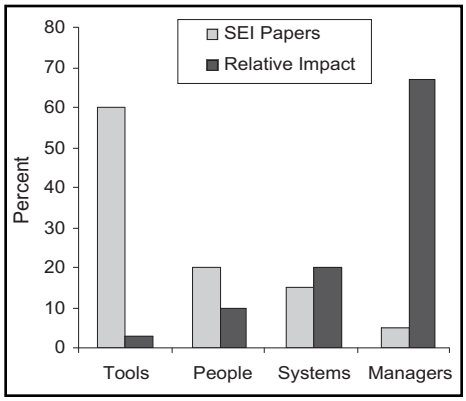
Over years of observing team communication and management issues, we have found four practical commandments that profoundly affect productivity. The four commandments deal directly with communication and collaboration effectiveness. The fourth commandment also addresses motivational and team issues, as well as a lack of continuity when members of the team are not available at all times. Since effective communication is the backbone of the discussion, we begin with a foundation in communication mechanics.

Mechanics of Communication

Broadly defined, communication means the act or process of communicating, and a process by which information is exchanged between individuals through a common system of symbols, signs, or behaviors. A related definition for collaboration is to work jointly with others or together, especially in an intellectual endeavor. Both elements are necessary to produce a software product.

Communication or information trans-

Figure 1: *Number of Publications Versus Productivity Impact*



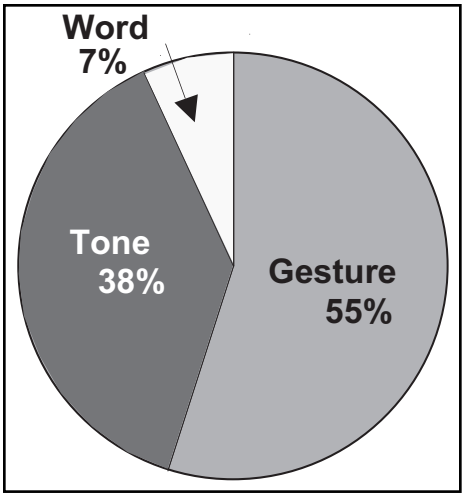
fer is one of the most important considerations in the world of productivity improvement. It dominates a large percentage of the time devoted to software development whether information is transferred via reports, analysis, problem resolution, or training. Several studies suggest the time spent in some form of communication exceeds 33 percent of a programmer's workday. Improved productivity, therefore, relies on the effective and efficient transfer of information.

Information Convection

In his book "Agile Software Development" [4], Alistair Cockburn described communication by comparing it to the dispersion of heat and gas. The concept is easy to apply to the dynamics of communication in the software development environment. *Convection currents* of information move about a work area just like the movement or dispersion of heat and gas. Air moves freely through an area unless the air is blocked or diverted by an obstruction.

Information moves in precisely the same fashion. When two programmers are seated at adjacent desks, they can discuss mutual problems freely and information flows unobstructed between the two people. The information flow, however, decreases as the programmers' separation distance increases. If a barrier or wall, real or perceived, is placed between the programmers, the information flow is further attenuated except for the information dispersion that occurs over the wall. If the programmers are placed in private offices, the information flow is blocked and becomes zero. Thus, instead of a communicated team effort, the programmer's attitude becomes, "I do my part and then throw it over the wall."

Figure 2: *Transfer of Information in Presentations*



Radiation

Information is also *radiated*. Radiation primarily occurs either aurally or visually. But it can also occur on a smaller scale from touch and smell. Information can also be radiated from whiteboards, paper, posters, sticky notes, and pictures. Because we want to maximize the amount of useful information being conveyed, we will discuss the optimal ways that information is radiated.

We will begin with close proximity communication and discuss the radiation sources one at a time. The optimal source of radiation communication is both voice and visual. Voice and visual communication is radiated by expression, gestures, pitch, volume, inflection, exaggerations, and movement. Two people discussing a problem at a whiteboard or at a computer terminal exemplify this ideal situation. This source of radiated information is optimal because of the response time between the speaker's statements and the listener's responses. The real-time nature of the conversation allows instantaneous questions to remove any misunderstandings and to clarify statements and questions.

The effectiveness of voice or visual radiation is supported by a well-known research study by Mehrabian and Ferris [5]. According to Mehrabian and Ferris, 55 percent of information in presentations is transferred by body language, i.e., posture, gestures, and eye contact (see Figure 2). Thirty-eight percent of the information is transferred through vocal tonality, i.e., pitch, volume, etc. Seven percent of the information transferred comes from the words, or content, of the presentation. These results are hardly surprising given that our body cues often convey the meaning of our words. For example, we all express many shades of meaning with the word *no* in normal conversation without giving much thought to the tone and body language accompanying the word.

The effectiveness of the information transfer, however, is diminished when we remove any source of radiation. For example, we can remove the visual part of the transfer by forcing the communicators to use a telephone. This eliminates all of the gestures, body language, and eye contact from the conversation. These important radiation sources are no longer available to reinforce understanding between the two individuals, and can lead to gaps in communication as well as misunderstandings. For example, we may change our language style when talking on the phone. This could lead to an inference of disinterest that seeing body language would dispel. People cannot see you nod your head in agreement on the telephone.

The information transfer is further diminished if we also eliminate the subtle elements of a conversation radiated by volume, tone, sarcasm, or disappointment by using e-mail instead of the vocal conversation. Think of the times you may have called or been called by someone about a date or an appointment and they made an excuse about not being available. The loss of vocal tone may cause you to miss the *get lost* message they are trying to convey.

By removing all radiating sources of information, finally, information transfer is significantly degraded when we remove the ability to respond and ask clarification questions by communicating solely on paper. We lose not only the subtle elements of our voice communication, but also the real-time element of the conversation necessary for feedback from one to another. Feedback may still be present, but at a much slower rate. This impairs the integrity or accuracy of the feedback as well.

Paper is good for formality and structure, but very limiting for information transfer.

Drafts

In the convection paradigm, a *draft* is a flow of unwanted information.

Ultimately, information flow occurs whether or not information transfer is desired. Two people sitting within the range of a *radiator* pick up information even when they are not directly communicating. The receiver can, and often will, respond to the radiator if the information is related to a topic of interest. Remember the E.F. Hutton commercial? When the information is important to someone; they listen. The receiver may also respond to the radiator if the information is disruptive. How many times have you asked someone to turn down the television or radio?

Now that we have established the communication analogy, let us look at the four communication commandments for efficient, effective software development.

I. Thou Shalt Not Construct Communication Barriers

As explained, walls impede the flow of information. Consequently, walls decrease productivity. This impediment includes both visible and invisible walls. Private offices and cubicles raise visible walls. Assume a large open area filled with workstations that are spaced 10 feet apart, front-to-back and side-to-side. People can move freely about the workspace. Since they are not totally enclosed, communica-

tion between individuals in this matrix should be reasonably unimpeded [6]. This was the original cubicle concept.

But we raise invisible walls if we alternate rows in this matrix with personnel from another project. This spacing causes the distance between related people to increase from 10 to 20 feet. This increased spacing between members of the development team decreases information flow. Thus, the presence of unrelated people forms a literal wall that impedes the information flow. The same effect can be achieved by randomly placing people from a second project. The information radiated by people from the unrelated second project creates what Cockburn referred to as a draft – a flow of unwanted information.

Invisible walls are also raised by increasing the space between every third row, so as to create an aisle between the rows. Thus, the aisle acts as a barrier or pseudo-wall. The aisle significantly inhibits the flow of information because people are naturally resistant to communication across assumed walls.

The modern technological solution to communication barriers is e-mail and network communications. This solution has been posed for local communication support and to justify remote software development teams. Ironically, this technological solution raises greater barriers than the cubicle example. Where people have at least some physical contact when in adjacent cubicles, remote locations are sometimes separated by a thousand miles. The loss of visual and voice radiation, as well as real-time responsiveness creates a virtual wall.

Skunk Works

A classic example of effective information convection is the Lockheed Skunk Works [7], primarily because it dispenses with both physical and non-physical walls. The most successful software organizations have followed this paradigm in the organization of their development teams and environments.

The Skunk Works is an unofficial name given to the Lockheed Advanced Development Projects Unit, which was the home of the legendary Kelly Johnson and his production team. In makeshift quarters, Johnson's team developed the U.S. Air Force's first operational jet fighter, the P-80 *Shooting Star*, in only 143 days. Since then, a number of famous aircraft, including the U-2, the SR-71, and the F-117 have been developed by this production unit. The newest Skunk Works project is the F-35 Joint Strike Fighter.

As a generic term, *skunk works* dates back to the 1960s. The common skunk works definition is a small group of experts who move outside an organization's mainstream operations to develop a new technology or application as quickly as possible, without the burden of the organization's bureaucracy or strict process application. Conventional skunk works operations are characterized by people who are free thinkers, creative, and who do not let conventional boundaries get in the way. The skunk works workspace is a physically open environment that encourages intra-team access and communication. Tools and processes are tailored and adapted to the project's requirements. Johnson established 14 Basic Operating Rules [7] to minimize development risk while maintaining the greatest possible agility and creativity in a lean development team. The rules covered everything from program management to compensation, and are relevant for any advanced research unit within a larger organization.

The management and teaming characteristics of the skunk works are important to our discussion of the commandments for a productive development organization primarily because they removed the walls or barriers that hamper communication.

Cube Farm

A counter-example to the skunk works approach to software development is the common cube farm. The cube farm violates all the rules for a productive environment in terms of both communication and collaboration primarily because they raise all the barriers that block communication. Unfortunately, the cube farm is the most common or widely used software development environment. Probably 90 percent to 95 percent of the development organizations operating today work in cube farms. A common programmer response when asked about their workspace is, "Scott Adams used our organization as the pattern for Dilbert." Many think Scott Adams is an alias for one of their employees.

In fact, the evolution of the cube farm, a grouping of cubicles that optimizes the number of people per square foot of floor space, did not begin as depicted in the Dilbert cartoons. In the late 1950s, typical offices were large open spaces filled with orderly rows of desks, and surrounded by private, closed offices for supervisory personnel. At about the same time, the Henry Miller Company approached Robert Probst [8], a professor of fine arts at the University of Colorado, to create a furniture design that would improve communication and productivity. The result, the Henry Miller Action Office system, appeared in the mid-60s. The approach started with a large open area, sectioned to give workers semi-private to private enclosed spaces where needed, but the work area was arranged in a way to provide ease of worker-to-worker and worker-to-manager interaction. The design promoted communal space for interaction. The Action Office was an immediate success.

Enter now the facilities planner or *space police*. Their plan was to remove all of the wasted open space to maximize the use of a building's floor space. Or, in other words, maximize the number of people per power outlet. The resulting cube farm does just that by providing high human density, easy reconfiguration, and facility cost savings. But the saved space is more than counteracted by the resulting high price in loss of product development efficiency and productivity. Thus, decisions by facility planners have dramatically affected project schedules.

This is because the cube farm, as it exists today, virtually eliminates information convection by blocking all, or essentially all, personal interactions. The standard six-foot by eight-foot sound insulated cubicle lacks space for a two-person discussion, contains no whiteboards or other communication media, and pipes drafts (white noise) into the farm background to suppress any information that might escape into the environment. In short, the cube farm is the least likely of all facility arrangements to encourage improvements in productivity.

II. Thou Shalt Dedicate the Project Area

The physical project area should be allocated to a specific development task and not shared by multiple projects. From the standpoint of information convection, all of the information moving about the development area should be related to the same software development activity. Mixing projects in a specified area creates drafts. The drafts are created by mixing people from unrelated tasks. Dedicating a specific project area places all of the development personnel in close proximity with as few sources for drafts as possible. Adding people from non-related projects also separates project-related people, thereby limiting the information flow and inhibiting discussion and collaboration.

Another side effect of an undedicated project area is that the presence of people from another task prevents the team from forming into a focused, cohesive unit. An

extreme view of this phenomenon occurs when the project area is a general software engineering area accommodating multiple projects. Project teams never form in this situation.

Several years ago, a software team carried the dedicated workspace concept to a new level. The manager physically moved the team to an unused cafeteria. With soap, water, and the support of Canteen Corp. (the vending machine supplier), the team created a project area remote from outside interference. The members were experienced, but not individual superstars. The team evaluation at project completion received the highest performance rating (aka, Seer [9] basic technology constant) recorded at the time.

A corollary to the second commandment is that *outsiders* should not mess with the project area. The project area needs to be the project domain, controlled by the project team.

III. Thou Shalt Provide Utensils for Creative Work

When we consider tools for creative work, we usually think of the technology bucket. Technology-oriented tools include programming languages, computer systems, development environments, CASE and scheduling tools, and formal practices and procedures. All of these *technology* tools affect productivity, but, as stated, this impact is minor compared to the productivity impact of poor communication.

We have learned from experience and research that communication and collaboration are key to productivity and quality improvement. Our earlier discussion about information convection and radiation suggests that a completely different set of low-tech utensils are best for creative work. These utensils include the following:

- Whiteboards.
- Easel pads.
- Butcher paper.
- Post-it Notes.
- Kitchenette (break room with whiteboards).
- Informal discussion areas (brainstorming area).
- Popcorn.

None of these utensils fit well within a cubicle environment. Whiteboards, Post-it Notes, and popcorn can be physically placed in a cubicle, but for *individual* use only. Group activities using the above utensils require large cubicles that support teams rather then separate them. The space police look at team space as wasted and they want to pack more individual bodies into that space. But, the environment we want to foster is people working together effectively. Effective team activities require utensils to support communication. You cannot tell a child not to eat with his or her hands without providing an alternative. Likewise you cannot build project teams without providing team-building tools.

When evaluating an organization's productivity, the presence or absence of these tools profoundly affects the result. Popcorn may seem like a strange tool, but it is almost always present in a highly productive work area. While popcorn does not analytically fit into any criteria for improved productivity, its odor seems to attract the necessary collaboration and enhanced communication. The scent of popcorn is indicative of people working together.

IV. Thou Shalt Not Share Resources

People-sharing between projects makes it impossible to form a genuine development team for any specific task. Part-time commitment does not permit shared individuals to fully participate in a task. This ultimately limits support physically as well as socially. Teams are sensitive to part-time participation. The part-time individual is, in a sense, an outsider and will not be trusted to carry out a task if not fully committed. Teams cannot gel when full-time participation in the project is not the norm.

Another phenomenon occurs when people are shared between two or more projects. Information that relates to one project becomes a draft for resources participating in a second project. Thus, when more than one project is active in a given area, the need for individual privacy becomes an issue due to the distracting information flow or noise in the area.

Food for Thought

Communication and collaboration are vital elements of the software development activity. When we accept this as a truth, we recognize the importance of making communication effectiveness a priority in project planning.

There are some important issues related to the success of environments associated with the four development environment commandments described in this article. The issues are the following:

1. The software development industry has pursued many technology approaches for improved productivity over the past 40 years. Communications and collaboration issues cannot be resolved with the next silver bullet (new technology tool).
2. Management culture changes slowly, if at all. (We learn from experience that we do not learn from experience.)
3. Most managers are not brave enough to keep their hands off, to accept that a major part of their business will be performed by a remote operation that they cannot interfere with. A large part of Lockheed's management resented the existence of the skunk works and its success.
4. Low team-staffing levels and efficient (highly productive) operations equate to low profits on traditional government contracts, which reward effort rather than results.
5. Small, highly cohesive teams having little interaction with the larger organization equates to little opportunity for raises and promotions, especially in a traditional organization that wants to reward managers based on the number of people supervised rather than on results.
6. The skunk works model, where the skunk works has considerable freedom to innovate and arrive at its own solution to the customer's problems, does not work well with customers or managers who want total control. The skunk works gives the team the power to create, communicate, and overcome challenges without micro-management.

There are management ideas to help enable the team communication and the project to succeed. Management style is inherently important in this promotion of team development by enhancing communication. The following summarizes this:

1. Management cannot be a bottleneck of communication. Management must allow the team to contact the necessary people both inside and outside the team to get the needed information.
2. Teams are not just created; they grow through communication, interaction, and trust. Management must recognize this and try to create not only membership in a team but an environment conducive to communication and interaction. After the membership and environment are in place, the trust grows. As part of the trust environment, team members need to feel that sharing information with others does not threaten their individual job status, ability to advance, or bonuses.
3. Managers, customers, and teams need an atmosphere of trust and accountability.
4. Management needs to view itself as support personnel that enable the team to succeed and not as the dictat-

ing, governing body.◆

References

1. Weinberg, G. Quality Software Management Vol. 3. Englewood Cliffs, N.J.: Prentice-Hall, Inc.: 15.
2. Boehm, B.W. Software Engineering Economics. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981: 486.
3. Standish Group International. The Chaos Report (2004). West Yarmouth, MA: Standish Group International, 2005.
4. Cockburn, Alistair. Agile Software Development. Indianapolis, IN: Pearson Education, Inc., 2002: 77.
5. Mehrabian, A., and S.R. Ferris. "Inference of Attitudes From Nonverbal Communication in Two Channels." Journal of Counseling Psychology Vol. 31 (1967): 248-52.
6. Becker, F., and W. Sims. "Workplace Strategies for Dynamic Organizations." Offices That Work: Balancing Cost, Flexibility, and Communication. New York: Cornell University International Workplace Studies Program (IWSP), 2000.
7. Janos, Leo, and Ben R. Rich. Skunk Works: A Personal Memoir of My Years of Lockheed. 1st ed. Back Bay Books, 1996.
8. Abraham, Yvonne. "The Man Behind the Cubicle." Metropolis Magazine Nov. 1998.
9. Jensen, R.W. An Improved Macrolevel Software Development Resource Estimation Model. Proc. of the Fifth International Society of Parametric Analysts Conference, St. Louis, MO. Chandler, AZ: ISPA, 26-28 Apr. 1983.

About the Authors

Randall Jensen, Ph.D., is a consultant for the Software Technology Support Center, Hill Air Force Base, Utah, with more than 40 years of practical experience as a computer professional in hardware and software development. For the past 30 years, he has actively engaged in software engineering methods, tools, quality software management methods, software schedule and cost estimation, and management metrics. Jensen retired as chief scientist of the Software Engineering Division of Hughes Aircraft Company's Ground Systems Group. He founded Software Engineering, Inc., a software management-consulting firm in 1980. He developed the model that underlies the SAGE and the Galorath Associates, Inc.'s SEER-SEM. He received the International Society of Parametric Analysts Freiman Award for Outstanding Contributions to Parametric Estimating in 1984. He has a doctorate in electrical engineering from Utah State University.

Software Technology Support Center
6022 Fir AVE BLDG 1238
Hill AFB, UT 84056
Phone: (801) 775-5742
Fax: (801) 777-8069
E-mail: randall.jensen@hill.af.mil

Leslie Dupaix is an engineer at the Software Technology Support Center (STSC) at Hill Air Force Base, Utah. He has been with the STSC since 1988 where he has worked embedded computer issues, including Ada and JOVIAL issues. He has been a certified Personal Software Process℠ instructor since 1997. Prior to STSC, Dupaix worked for the Air Force programming operational flight programs and as the government language expert during which time he was on the JOVIAL language control board and the U.S. Air Force Ada insertion team. He has worked on reviews for the F-16, global positioning system, C-17, Advanced Tactical Air Reconnaissance System, LOROPS, the Air Force 1750A Ada compiler system, and other systems. Dupaix has been a member of the Department of Defense's Ada Software Engineering Education and Training Team since 1988. He has a Bachelor of Science in electronic engineering from Brigham Young University, Utah.

Software Technology Support Center
6022 Fir AVE BLDG 1238
Hill AFB, UT 84056
Phone: (801) 775-2064
Fax: (801) 777-8069
E-mail: les.dupaix@hill.af.mil