Laziness, Impatience, Hubris:
Personality Traits of a Great Programmer

John E. Bentley, Wachovia Bank, Charlotte NC

ABSTRACT

Not everyone can be a great computer programmer. Many don't really have the desire, some lack a higher-level technical aptitude, and still others don't have the personality needed to be great. Wait, you say. What's personality got to do with it? Well, most programmers don't have a colleague who isn't intelligent enough to do the work, but most <u>do</u> know someone who isn't exactly temperamentally suited to the job. To help explain the impact of personality on success as a programmer, this paper will present some of the traits that can be indicators of professional success or failure in the field. For one, a sense of humor is important because the computer "doth make fools of us all."

INTRODUCTION

The importance and the difficulty of matching the right person with the right job is such common knowledge that it shouldn't have to be said. A literature search of how to match people with jobs will return a seemingly endless list of references from both the academic and popular press. Probably tens of thousands of well-educated corporate employees and independent consultants make their living matching people with jobs. With all these resources available, why is hiring the right person so difficult to do? It could be that the people making the actual hiring decisions—the first and second-level team leaders and managers—are not really trained how to spot the right person and, just as importantly, how to spot the person who isn't right for the job. Think about it. Have you ever interviewed someone for a job? Even if you don't make the final hiring decision, have you ever spoke with a candidate and then provided your opinion about their qualifications and capabilities? What training do you have?

From this author's experience, too often SAS professionals are asked to participate in the employee selection and interviewing processes without being adequately prepared to do so. In these cases, the resulting interview might more closely resemble a rather stilted conversation during which the interviewer asks a standard set of questions ("Tell me about a time when you had to work under severe time constraints.") and then listen to responses that sound a bit like war stories. These standard questions are intended draw out the candidate's intelligence, training, experience, and capabilities as they relate to the job. While these are important and critical attributes and the standard questions can be useful for objectively discovering them, what about the candidate's personality?

It's generally agreed that personality is a <u>critical</u> factor in determining job success. No matter how educated or experienced a programmer is, without certain personality traits they're likely to achieve a lesser degree of success if not outright failure. Are there any questions on the list intended to objectively learn about personality or does the interviewer have to go with their subjective 'gut' feeling?   Maybe yes, maybe no. But because interviewers are less likely to objectively examine the candidate's personality, they're more likely to make a personality mistake in hiring than they are an intelligence mistake.

A few IT researchers and writers have looked at personality and identified certain personality traits that they argue can be used as early indicators of success or failure in the field of computer programming broadly defined. Gerald Weinberg (1998) goes so far as to propose that personality is more important than intelligence because there are few people working in the field who are not intelligent enough to program (they'd soon be not working in the field) but there are lots of people whose personality is such that they really should be doing something else. Using work done by Weinberg and others, this paper will first look at some of the classifications of personality, such as the Myers-Briggs typology, and then discuss specific personality traits that seem to be critical for higher levels of success as a programmer.

THE PROGRAMMER STEREOTYPE

> Nerd (nûrd) *n. slang:* A person who is single-minded or accomplished in
> scientific or technical pursuits but is felt to be socially inept.
> *dictionary.com*

The stereotypical programmer is a shy young man, either scrawny or overweight, who works by himself in an 8'x8' cubicle in a bigger room of dozens cubicles, each holding someone just like him. He intensely concentrates on writing cryptic instructions to coax a computer to do what is needed. He can concentrate twelve to sixteen hours at a time and often loses track of time and works through the night to realize what he perceives as an artistic creation. He devotes his evenings, weekends, and summers to work. He subsists on pizza, Twinkies, and Mountain Dew. When

interrupted unexpectedly, the programmer may respond with strings of gibberish—"SET PROC DATA STEP LIBNAME RUN!" He has no social life and any hobbies he may have resemble his work. The programmer breaks away from the computer only to attend Star Trek conventions and maybe watch Monty Python reruns. In some companies he is regarded as an indispensable genius; in others he is tolerated as an eccentric artist. Vital information is stored in his head and his head alone. He is secure in knowing that even though his job is critically important, few people compete for it. What he doesn't realize is that few people want it. (McConnell, 1999)

A few years ago, *USA Today* reported that the techie nerd stereotype is so well entrenched that students in high school ranked computer jobs near the bottom of their lists of career choices. *The Wall Street Journal* reported that film crews have difficulty presenting stories about leading-edge software companies in an interesting way because every story starts with "an office park, a cubicle, and a guy sitting in front of a keyboard with a box on his desk." Even within the profession the stereotype is fostered. The associate director of Stanford University's computer science program was quoted by the *New York Times* as telling incoming students that software jobs are "mind-numbingly boring."

How much of this is true? In general, not much. Here's someone from the Techie Geek and Nerd Hall of Fame.

Dr. Mary Grace Hopper, RAdm USN, Ret.

1928—BA in Mathematics and Physics, Phi Beta Kappa, Vassar
1934—PhD in Mathematics, Yale,
1942—Programming team member for the Mark I, the worlds first automatically sequenced digital computer
1950—Developed the first compiler, A-0, for translating symbolic mathematical code into machine code
1954—Developed the B-0 compiler for business tasks
1956—Taught UNIVAC I and II to understand twenty English-like statements
1959—Directed the team that designed the COBOL language
1969—Named the first "Computer Science Man-of-the-Year"
1973—First female Distinguished Fellow of the British Computer Society
1986—Retired as Rear Admiral, United States Navy
1991—Awarded the National Medal of Technology

"It's always easier to ask forgiveness than it is to get permission."

PERSONALITY TRAITS

Traits are lasting aspects of individuality that differentiate one person from another. All persons share the same basic set of traits, but people are different in how much the trait applies to them. For instance, everyone has a "friendliness" trait, but some people are high in "friendliness" and some are low in it.   Gordon Allport (in Robert Feldman, 1992) proposes that a small set of *central traits* make up the core of a person's personality. *Secondary traits* are always present but are less important in defining one's personality.

So what are these traits that can be either central or secondary? As with most issues related to human psychology, personality experts disagree on what they are and, where they do agree, often disagree on how important each is in defining the same person's personality. One group of experts say that there are sixteen core personality traits, and include things like intelligence, desire for adventure, practicality, and confidence in their list. Another group says that there are only seven core traits, and yet another group says there are only two and neither of them is in the group of sixteen. Despite the proliferation of various schools of thought, there does seem to be general consensus that five traits form a core personality: agreeableness, conscientiousness, extroversion, intellect, and neuroticism. The first four are self-explanatory, but neuroticism is the degree one possesses a "mental and emotional disorder that affects only part of the personality, is accompanied by a less distorted perception of reality than in a psychosis, does not result in disturbance of the use of language, and is accompanied by various physical, physiological, and mental disturbances." (*dictionary.com*)

It's important to keep in mind that traits are really just labels for comparing one person with another. One person may seem to be more "friendly" than another, but people behave differently depending on the situation. A person who is "shy" around strangers could be "very friendly" when with a group of people she knows.   Also, personalities are known to change over time, sometimes rather suddenly in response to external events such a marriage.

PERSONALITY TESTS AND TYPES

Ever since we realized that there are different personalities there have been ways of assessing personality and grouping people of similar personalities. Almost everyone is familiar with perhaps the earliest method—astrology, in which one's personality is determined by the time and place of their birth. Although generally discredited, it can sometimes seem to be eerily accurate.

Modern personality assessment methods are based on tests developed by psychologists to dentify and measure traits and behaviors that the person possesses. The tests range from self-reported survey instruments to question and answer sessions with a therapist to direct observation of a person's daily activities. These tests can't tell everything, but many of them have been refined to the point where their conclusions are generally accepted as accurately describing a person's personality and some are admissible in courts of law.

One common clinical test that is generally not used in job candidate evaluation but is accepted in legal situations is the Minnesota Multiphastic Personality Inventory-2 (MMPI-2). This self-administered test is a revision and extension of the original MMPI developed in the 1930s. The complete battery of tests has almost six hundred statements about different aspects of personality, and the subject is asked to indicate whether the statement is true or false as it applies to them. The result is a self-described profile of the person's personality characteristics as well as any psychological problems and symptoms that may exist. Personality is measured on ten clinical scales, six validity scales (to identify response inconsistencies), and a host of supplementary scales. The clinical scales and subscales measure things like tendency to have social contact, general social adjustment, level of paranoia, and response to stress. Patterns and relationships between scores on the scales are used to define a personality.

There are a dozens of tests designed to identify the career and educational choices appropriate to specific personalities. One of the oldest is the 1941 Primary Business Interests Test. It "is designed to measure an individual's preferences for the specific job activities which characterize beginning business occupations."   A more recently-developed test is the Jackson Vocational Interest Survey (JVIS), which since 1969 has been used by high-school and college students to guide them in vocational and educational decisions. The JVIS consists of 289 pairs of work-related activities and the subject chooses the activity that they like the most or dislike the least. For example, one pair might be "Raising turkeys" and "Compiling definitions for an accounting textbook". The test is now administered by computer and test pairs are dynamically assigned based on recognized patterns of responses.

Perhaps the best known and most widely accepted personality test is the Myers-Briggs Type Indicator (MBTI) personality inventory. The Center for Applications of Psychology estimates that about 2 million people each year take the MBTI test. The starting point for the MBTI is psychologist C.C. Jung's work from the 1920s in which he theorized that much seemingly random variation in behavior is actually quite orderly and consistent because it is due to basic difference in the way individual use their perception and judgment. Jung's theory identified four dichotomies based on perception and judgment, and since the 1940s the MBTI has been used to classify personalities into one of sixteen types based on these dichotomies.

Table 1—Jungian Personality Dichotomies

Jungian Dichotomy	Description	Myers-Briggs Type Indicators
Favorite World	Do you prefer to focus on *the outer world* or on *your own inner world*?	Extroversion (E) or Introversion (I)
Information	Do you prefer to focus on *the basic information you take in* (facts) or do you prefer to i*nterpret and add meaning*?	Sensing (S) or Intuition (N)
Decisions	When making decisions, do you prefer to *first look at logic and consistency* or *first look at the people and special circumstances*?	Thinking (T) versus Feeling (F)
Structure	In dealing with the outside world, do you prefer to *get things decided* or do you prefer to *stay open to new information and options*?	Judging (J) or Perceiving (P)

After a person takes the MBTI test, one letter from each of the Type Indicators is used to describe the person, resulting in an acronym like ESFP. These letters indicate an individual's personality tendencies or preferences. They don't indicate how a person will act in a particular situation or circumstances. For example, a person with a preference for Introversion may have purposefully developed their extroversion so they can be more effective in a business setting that requires them to make formal presentations.

The sixteen personality types defined by the Myers-Briggs Type Indicator instrument are often shown a "Type Table" that contains descriptions that read like something from Omar the Astrologer. The combinations of initials are based on which sides of the four dichotomies the subject leans toward. Tieger (2001) presents detailed information on interpreting and using MBTI for career planning and development.

One commonly accepted application of MBTI typology is that if everyone knows their team member's personality type and bases their interactions on them, then group dynamics will be more effective. More importantly for this paper, at least at face value these descriptions indicate that some personality types may be more appropriate for some lines of work than others—so a candidate's MBTI would be useful to know during a job interview.

Table 2—MBTI Personality Types

ISTJ: Quiet, serious, earns success by thoroughness and dependability. Practical, matter-of-fact, realistic, and responsible. Decide logically what should be done and work toward it steadily, regardless of distractions. Take pleasure in making everything orderly and organized. Value traditions and loyalty.	**ISFJ**: Quiet, friendly, responsible, and conscientious. Committed and steady in meeting their obligations. Thorough, painstaking, and accurate. Loyal, considerate, notice and remember specifics about people who are important to them, concerned with how others feel. Strive to create an orderly and harmonious environment.	**INFJ**: Seek meaning and connection in ideas, relationships, and material possessions. Want to understand what motivates people and are insightful about others. Conscientious and committed to their firm values. Develop a clear vision about how best to serve the common good. Organized and decisive in implementing their vision.	**INTJ**: Have original minds and great drive for implementing their ideas and achieving their goals. Quickly see patterns in external events and develop long-range explanatory perspectives. When committed, organize a job and carry it through. Skeptical and independent, have high standards of competence and performance.
ISTP: Tolerant and flexible, quiet observers until a problem appears, then act quickly to find workable solutions. Analyze what makes things work and readily get through large amounts of data to isolate the core of practical problems. Interested in cause and effect, organize facts using logical principles, value efficiency.	**ISFP**: Quiet, friendly, sensitive, and kind. Enjoy the present moment, what's going on around them. Like to have their own space and to work within their own time frame. Loyal and committed to their values and to people who are important to them. Dislike disagreement and conflict; do not force their opinions or values on others.	**INFP**: Idealistic, loyal to their values and to people who are important to them. Want an external life that is congruent with their values. Curious, quick to see possibilities, can be catalysts for implementing ideas. Seek to understand people and to help them fulfill their potential. Adaptable, flexible, and accepting unless a value is threatened.	**INTP**: Seek to develop logical explanations for everything that interests them. Theoretical and abstract, interested more in ideas than in social interaction. Quiet, contained, flexible, and adaptable. Have unusual ability to focus in depth to solve problems in their area of interest. Skeptical, sometimes critical, always analytical.
ESTP: Flexible and tolerant, they take a pragmatic approach focused immediate results. Theories and conceptual explanations bore them – they want to act energetically to solve the problem. Focus o n the here-and-now, spontaneous, enjoy each moment that they can be active with others. Enjoy material comforts and style. Learn best through doing.	**ESFP**: Outgoing, friendly, and accepting. Exuberant lovers of life, people, and material comforts. Enjoy working with others to make things happen. Bring common sense and a realistic approach to their work, and make work fun. Flexible and spontaneous, adapt readily to new people and environments. Learn best by trying a new skill with other people.	**ENFP**: Warmly enthusiastic and imaginative. See life as full of possibilities. Make connections between events and information very quickly, and confidently proceed based on the patterns they see. Want a lot of affirmation from others, and readily give appreciation and support. Spontaneous and flexible, often rely on their ability to improvise and their verbal fluency.	**ENTP**: Quick, ingenious, stimulating, alert, and outspoken. Resourceful in solving new and challenging problems. Adept at generating conceptual possibilities and then analyzing them strategically. Good at reading other people. Bored by routine, will seldom do the same thing the same way, apt to turn to one new interest after another.
ESTJ: Practical, realistic, matter-of-fact. Decisive, quickly move to implement decisions. Organize projects and people to get things done, focus on getting results in the most efficient way possible. Take care of routine details. Have a clear set of logical standards, systematically follow them and want others to also. Forceful in implementing their plans.	**ESFJ**: Warmhearted, conscientious, and cooperative. Want harmony in their environment and work with determination to establish it. Like to work with others to complete tasks accurately and on time. Loyal, follow through even in small matters. Notice what others need in their day-by-day lives and try to provide it. Want to be appreciated for who they are and for what they contribute.	**ENFJ**: Warm, empathetic, responsive, and responsible. Highly attuned to the emotions, needs, and motivations of others. Find potential in everyone and want to help others fulfill their potential. May act as catalysts for individual and group growth. Loyal, responsive to praise and criticism. Sociable, facilitate others in a group, and provide inspiring leadership.	**ENTJ**: Frank, decisive, assumes leadership readily. Quickly see illogical and inefficient procedures and policies, develop and implement comprehensive systems to solve organizational problems. Enjoy long-term planning and goal setting. Usually well informed, well read; enjoy expanding their knowledge and passing it on to others. Forceful in presenting their ideas

McConnell (1999) refers to two large studies that produced estimates saying that 20-40 percent of software developers had an MBTI personality type of ISTJ (introversion, sensing, thinking, judging) or INTJ (introversion, intuition, thinking, judging). Both of these types tend to be serious and quiet, practical, orderly, logical, and successful through concentration and thoroughness. Programmers are widely perceived as introverts, and MBTI statistics show that one-half to two-thirds are introverted compared to about one-quarter of the general population.

McConnell finds the MBTI S/N (sensing/intuition) and T/F (thinking/feeling) attributes particularly interesting because they are used to describe one's decision-making style. MBTI indicates that between 80 and 90 percent of programmers are T's, compared to about 50 percent of the general population. This is appropriate, considering that T's are more logical, analytical, dispassionate, and impersonal.

McConnell also says that programmers are about evenly split between Sensing and iNtuition, and the difference should be quickly apparent. S's are methodical, precise, and concrete. They like to specialize and develop a single idea in depth rather than explore many paths. An archetypical S programmer would be an expert with a very detailed knowledge of a specific language or technology. An N programmer, on the other hand, is a generalist who is familiar with a wide range of tools and considers the merits of many possible solutions before selecting one. In extreme cases, they could drive each other crazy if forced to work together—S focuses on the technical details before N has finished exploring the alternatives. N moves from one idea or possible solution to the next before S feels they have explored it in sufficient depth.

Tieger (2001) agrees with McConnell that programming (and most technical jobs) is a good fit for an INTJ personality, but doesn't agree that ISTJ is a highly-desirable profile. The ISTJ may have some trouble 'thinking outside the box' to the degree a programmer must when devising a creative solution to a problem. Tieger also believes that programming would work for ISTP personality. The main difference between INTJ and INTP is the way that they assimilate and act upon new information. The former is more likely to jump right in and start working while the latter deliberates and reflects before beginning.

In fairness, we must note that there is a great deal of criticism of the MBTI. Much of it stems from criticism of the Jungian personality type theory that provided the theoretical foundation of MBTI, and other criticism is derived from the fact that neither of the originators of MBTI— Katherine Cook Briggs and her daughter Isabel Briggs Myers—were trained psychologists. Katherine did not attend college and Isabel held a Bachelors degree in Political Science. Most troubling perhaps is that quite a few studies show that when retested, even within five weeks, as many as 50 percent of people will be classified into a different personality type.

THE "PROGRAMMER PERSONALITY"

> "To find out about the beliefs and values of a software developer, look at their code."
> *Richard Bandler*

If we were to develop a personality filter to help us find successful programmers, what would be the traits we would let through? One place to start is by using common sense to list the traits we think a good programmer would have— from what's in the literature we wouldn't be far off the mark.

Table 3—Common sense programmer traits

|			
• logical	• curious nature	• tenacious	• patient
• self-confident	• detail-oriented	• methodical	• polite
• creative	• calm	• analytical	• cooperative
• rational	• ability to see the implications of a decision	• desire to solve problems	• seeks mental challenges
• self-motivating	• ability to focus	• works well alone	• focused

Next we might want to look at a study done in 2004 by two researchers at the Washington State University. They speculated that gender segregation and turnover of women in information technology jobs is due in part to the perception of IT as a "masculine" occupation. (Joshi and Kuhn, 2004.) Using study subjects from a large international consulting firm, the researchers identified 752 personality traits and behaviors. Many of the traits shared by the "successful" analysts (not programmers) who participated in the study were not at all surprising. Some of them, though, were very surprising because they are so contrary to the MBTI traits. (Joshi and Kuhn have so far published only a preliminary summary of their results. The study data collection is still in progress.)

Table 3— Joshi and Kuhn's IT Success Traits

No surprise	Surprising	Very Surprising
Analytical Nature	Builds and maintains relationships	Competitive
Good at problem solving	Performs activities outside scope of responsibilities	Aggressive
Efficient		Acts as a leader
Cooperative		
Reliable		
Tactful		
Helpful		
Quick learner		

DILBERT'S PERSONALITY

Dilbert has the social skills of a mouse pad and he'd rather surf the Internet than Waikiki (which is a good thing, considering the body he developed after years of sitting in front of a PC screen).

Our common-sense traits and those from the Washington State
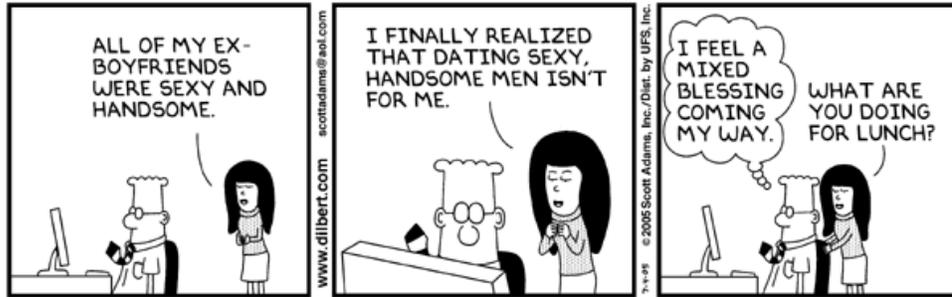
© Scott Adams, Inc./Dist. by UFS, Inc.

study are a bit too generic to provide a lot of insight into the 'programmer personality'. With tongue firmly in cheek, in *The Dilbert Principle* Scott Adams (1996) tells how and why IT workers are not like other people and tries to help non-technical people to understand them. Programmers and other IT professionals have distinctive personalities in that they

- Are socially inept and "normal" people who expect things like stimulating and thought-providing conversation with them will be disappointed.
- Struggle with a social life because they cannot place appearance above function. Body language, they believe, is too imprecise to be useful for communication.
- Are fascinated with gadgets. Generally speaking, non-technical people believe the adage "if it aint broke, don't fix it." IT professionals believe that if it ain't broke, it doesn't have enough features yet.
- Make clothing their lowest priority assuming the basic temperature and decency threshold has been satisfied. Anything more is a waste.
- Are excessively honest even in the most awkward situations, except for white lies like "I won't change anything without clearing it with you."
- Are exceedingly frugal because when it comes to money every spending situation becomes a problem of resource optimization.
- Delight in sharing their wisdom even in areas in which they have no experience. They believe that their command of logic provides them with inherent insight into any field. Non-technical people, of course believe that knowledge comes through experience.
- Have amazing powers of concentration. Of course this leads to devoting days to devising an elegant solution to a simple problem because they can hear the computer laughing at them.
- Hate risk because managers make such a big deal out of one little mistake. Just look at the *Hindenberg, Apollo 13,* the Hubble space telescope, and the Space Shuttle *Challenger*,

GERALD WEINBERG AND *THE PSYCHOLOGY OF COMPUTER PROGRAMMING*

In the classic study *The Psychology of Computer Programming*, Gerald Weinberg asserts that "personality is more important than intelligence in programming" because more programmers fail due to personality faults than due to lack of intelligence. Weinberg asserts that pre-selection via educational requirements prevents people with below-acceptable intelligence from entering the field. There is, however, no comparable process to screen on the basis of personality. One reason for that is because personality changes in response to environment and so can be somewhat transitory. The personality that graduates college may not be the same one that started four years earlier.

Weinberg uses personal experience and anecdotal material to identify traits that give an indication of success and failure as a professional programmer. First on his list is an ability to tolerate stressful situations for a week or more.

Two facts of a programmer's life are that schedules are imposed from the outside and requirements change. More than once, a job can literally hinge on finding and fixing a bug by tomorrow. There is always the potential shock of having an assignment cancelled half-way through or changed so much that the previous work is worthless but the schedule remains the same. People who are not flexible and adaptable to rapid change—not just change but rapid change—will not do well as a programmer.

Neatness is important. Neatness in this case meaning a "slight compulsion" for organizing notes, documentation, and other working materials. Recognizing that reference materials are needed to write even a moderately complicated program or application and the documentation that must (should?) accompany it, Weinberg notes one organization that has a writing test as part of its interview process—candidates are evaluated based on the neatness of what they turn in, not it's content.

Programmers need at least a small dose of humility and, at the same time, a bit of assertiveness. Without humility, the programmer will become overconfident which eventually leads to over-reaching and self-destruction. As can be seen in classical Greek drama set on a modern IT stage, human inadequacy is revealed in the novice programmer who, upon learning a few SQL techniques, fancies herself an expert but goes down in flames when she generates a Cartesian product that fills up all the work space on the production server. Assertiveness—force of character—is needed to get things done in any field and programming is no exception. Getting things done means moving around obstacles, jumping over them, or even knocking them down. This requires a degree of forcefulness, not acquiescence or complacency. Assertiveness and humility moderate each in a programmer's personality. Weinberg likens assertiveness to a steam boiler and humility to a safety valve. Without the boiler, no work gets done; without the safety valve, there's real danger of an explosion.

Weinberg's last personality trait for programming is a sense of humor and the ability to laugh at oneself. Anyone unable to share a laugh at their own foolish assumption, obvious mistake, or witless blunder will be unable to tolerate programming for any length of time. Everyone knows that the Programmer's Theme Song begins "aaaaahhhhhhhh". Only those with a programmer's personality know that the second stanza is "ha ha ha ha ha".

THE THREE GREAT VIRTUES

> "We will encourage you to develop the three great virtues of a programmer:
> laziness, impatience, and hubris."
> *Larry Wall, Programming Perl*

Laziness
> The quality that makes you go to great effort to reduce your overall energy expenditure. It makes you write robust, modular, well-documented programs so you can reuse the code.

Impatience
> The anger you feel when the computer is being lazy, which happens when another programmer is not lazy. It makes you write programs that use minimal code so they're fast, efficient, and anticipate what needs to be done.

Hubris
> The pride that makes you write and maintain programs that you and your peers will admire. If hubris is uncontrolled or undeserved, it can also get you in trouble.

Larry Wall chose these particular words only partly to be attention-getters. Other words could have been used, but these were selected both for their contrarian nature and descriptive ability. In a programmer's personality, when they exist as described they really are desirable. Of course, merely possessing these traits don't automatically qualify a programmer as great. As Brad Appleton (2004) notes, you've got to have the right quantity of each and they have to be mixed just right. They can also be used to describe a lazy, impatient, pompous ass.

Laziness is the mother of necessity, right? But it's also the mother of invention. (This is certainly not to suggest that Frank Zappa was lazy.) A lazy programmer works hard *now* so that in the *future* she won't have to work so hard. How many times have you hand-coded the same or a similar "one shot" task? Too many times, I'd guess. If you'd be lazy about it, you would have done it the first time in such a way that the code is easily recycled and adapted to similar tasks. Even lazier approach would be to see if anyone else has already solved your problem. Taking this a step further, a lazy approach to programming might involve the work of designing and implementing a code snippet filing system so that you and your team can quickly and easily share the code you need. Yes, it takes work to be lazy, but it's work now that will make life easier later.

So what we mean by lazy is "attempt to minimize the amount of labor it takes to get the job done". You want to minimize your own work and, possibly, minimize the work of your co-workers by writing code that can address multiple problems and be used in multiple situations. The guiding phrases are "write code once and only once" and "write

modular programs". Oh yes, and "document your programs" so you don't have to spend time and energy later figuring out what you did or why you did it.

Impatience is closely related to laziness. It has to do with how long it takes you to get frustrated enough to automate repetitive code and eliminate redundant code. It also has to do with your need for speed. Impatience helps bridge the vast gulf between good and great. How many SUGI papers on program efficiency do you have in your Tips and Clues binder? (You have one, right?) Do you use macros? If you do, do you write macro programs? Do you use arrays? Yes? But do you use temporary arrays? Do you use the SET statement WHERE option instead of a subsetting IF? You may use functions, but do you nest them regardless of how complicated it looks? Sure, brute-force code is easier to read (sort of), but that's what in-line documentation is for, right?

Dickerson (2002) equates impatience with vision—a sense that there's a better way to do everything. It's what drives the whole computer industry and provides the foundation for Moore's Law. Selectively apply the concept of "cheaper, faster, better" and you'll be a better programmer.

"Those whom the gods would destroy they first make proud." Hubris is often called a "tragic flaw"—exaggerated, undeserved pride or self-confidence that results in personal destruction. In Greek mythology, there is a goddess called Hubris (or Hybris) who personifies undeserved pride, overconfidence, insolence, and lack of restraint. Not surprisingly, she spends most of her time living among mortals.

In a programming context, the focus of one's pride must be pride in craftsmanship and the satisfaction of a job well done. Peer recognition may or may not follow. Programmers with hubris go above and beyond the business and technical requirements to make a product that excells at what it does. Quite often it may be great in ways that the customer doesn't care about or see—code formatting style and conventions; readability and documentation; flexibility and maintainability—but the programmer and his peers will know. Sure, these are all simply good coding techniques and practices, but programmer with hubris will take them much further than most do.

The extra efforts that lead to deserved hubris are not motivated by a desire to improve customer satisfaction or even to improve employer satisfaction. The primary motivation is the need for self-satisfaction and heightened self-esteem, along with the expectation that one's true peers will appreciate our talents. The motivation, then, can be seen as selfish—to produce something we can feel truly proud about.

Hubris is not necessarily a trait for which one consciously strives. It often somehow just creeps up over time as the result of a very high level of programming skills and lots of experience with many different assignments and situations. Great programmers are smart but know their limitations, can be tempermental but try to be a team player, and sometimes seem a bit self-righteous because they honestly think that their way is the best way. They're a challenge to manage too. If they're properly managed, though, they can make great contributions to corporate success and achieve great professional success and recognition.

In the wrong person, hubris will sooner or later lead to a public failure or even project disaster. Pathological problems they may have can include not sharing design information and hoarding souce code; claiming no time for code reviews and being insulted that someone would want to review their code; not adhering to corporate standards that they don't agree with. Eventually they will find out too late that they're not as good as they think they are and when they go down in flames they may take the entire project with them. Ego isn't a problem for programmers—egotism is.

CONCLUSION

> "I suspect that if I went to a job interview and said that laziness, impatience, and hubris
> are my three best assets, I wouldn't get the job."
> *John Bentley*

I think that a great programmer

- Knows that programming is a creative art and is anything but boring
- Takes great pride in his work and gets great satisfaction from it
- Tries to reduce the complexity of both the problem and the solution
- Is in a hurry but is never too busy to help others learn
- Seeks out constructive criticism and provides constructive criticism for others
- Has worked on failed projects but has consciously learned from those failures
- Is a master of his tools
- Never stops learning and gets a thrill from the 'aha' moments

Depending on the day of the week, I can be one of several MBTI personality types. Am I lazy because I'd rather not spend two days pounding out a thousand lines of code to solve a problem with brute force when I could spend a day designing an elegant five hundred line solution, a day writing and documenting it, and then another day deleting half of it to make what remained more robust and easier to maintain? Yes, I think I am. Am I impatient when someone on my team writes a thousand-line program that could be five hundred lines? Yes I am, because when there's a problem in a few months I'm going to get a headache helping them figure out what does what, finding the problem, and then rewriting the code to fix it. Do I have hubris? No, but give me a few more years.

REFERENCES AND RESOURCES

- Adams, Scott (1996) The Dilbert Principle. New York: HarperCollins.
- Appleton, Brad (2004) "Laziness, Impatience, Hubris" on the Wiki Wiki Web at
  http://www.c2.com/cgi/wiki?LazinessImpatienceHubris
- Dickerson, Chad (2002) "In Praise of Laziness" in InfoWorld, February 2nd.
- Feldman, Robert S. and Joel A. Fienman (1992) Who You Are. New York: Venture Books.
- Hunt, Andrew and David Thomas (2000) The Pragmatic Programmer. Reading, MA: Addison Wesley Longman, Inc.
- McConnell, Steve (1999) After the Gold Rush: Creating a True Profession of Software Engineering. Redmond, WA: Microsoft Press.
- The Skeptic's Dictionary at http://www.skepdic.com/myersb.html
- Tieger, Paul D. and Barbara Barron-Tieger (2001) Do What You Are. Boston: Little, Brow, and Company.
- Twerski, Abraham J. (1998) That's Not a Fault…It's A Character Trait. New York. St. Martin's Press.
- Vocational Tests at http://www.yorku.ca/psycentr/tests/voc.html
- Wall, Larry et al. (2000) Programming Perl, 3rd Edition. San Francisco: O'Reilly.
- Wall, Larry (1999) "Diligence, Patience, Humility" in Open Sources: Voices from the Open Source Revolution. San Francisco: O'Reilly.

AUTHOR BIOGRAPHY AND CONTACT INFORMATION

Since 1987 John Bentley has used SAS in the healthcare, insurance, and banking industries. He is currently an Assistant Vice President with Wachovia Bank's Corporate Data Management Group where he supports the Bank's data warehouse and data marts. John is a SAS Certified Advanced Programmer, has been a Section Chair at both SUGI and SESUG, and is past-Chair of the Charlotte Area Wachovia In-House SAS Users Group. His current professional interests are automating SAS programs and parallel computing. John has a Masters degree in Political Science with a Concentration in Southeast Asian Politics and is intermittently enrolled in a Masters of Information Systems program. Driven from Chicago by the weather, he hopes to never again live North of the Mason-Dixon Line.

John E. Bentley
Corporate Data Management and Governance
Wachovia Bank
201 S. College Street, NC-1025
Charlotte NC 28210
john.bentley@wachovia.com

DISCLAIMER