TITLE: "THEORY P: THE PHILOSOPHY OF MANAGING PROGRAMMERS"

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@attglobal.net
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"There are very few true artists in computer programming, most are just house painters."
- Bryce's Law

Over the last 100 years, three distinctly different theories of management have emerged: Theories "X", "Y" and "Z". All three are based on how management perceives the work force in terms of their intelligence level, motivation and attitude towards their job. Consequently, this perception becomes the basis for formulating formal policies and standard practices towards managing employees.

Although the delineation of "X", "Y" and "Z" represent totally different management philosophies, few companies will formulate a style of management based on a single theory. In reality, companies use various elements from all three theories based on different situations, everything from autocratic control to casual democracy.

The concept of Theory P does not attempt to introduce any new theories of management. Instead, it identifies those elements from Theories "X", "Y" and "Z" pertaining directly to the management of Programmers, hence the Theory "P" designation. Theory P, therefore, represents a style of management for a particular job segment.

Theory P was created in order to capitalize on the human resources responsible for developing and maintaining computer technology. In many cases, management is faced with a paradox: how to manage the programming department without irritating the programmers and cause them to abandon the company, leaving corporate systems prone to malfunction and in need of maintenance. Programmers are hip to this and often use this as leverage for job security. As such, corporate management is unsure how to properly manage this class of

(continued on page 2)

HOW DO WE MANAGE?		
THEORY	**BACKGROUND**	**CHARACTERISTICS**
THEORY X	Developed from time-and-motion studies by Frederick W. Taylor (19th century Industrial Engineer).	1. People have a natural aversion to work. 2. People need to be coerced to achieve goals. 3. Average person prefers to be directed, wishes to avoid responsibility, has little ambition, and wants security most.
THEORY Y	Developed from experiments at the Western Electric Hawthorne Works in Chicago (1930's). Management giving special attention to people resulted in improved performance.	1. Work is as natural as play or rest. 2. People will achieve goals they deem important. 3. Commitment/reward relationship. 4. People accept & seek responsibility. 5. People can use imagination & creativity. 6. More brain power is used.
THEORY Z	Developed by William Ouchi (UCLA) based on study of Japanese businesses during the 1970's. Observed higher productivity because Japanese society encourages mutual trust and cooperation.	1. Long term employment. 2. Employees need freedom to grow. 3. Group decision making. 4. Subordinates are whole people. 5. Management is concerned with welfare of subordinates. 6. Open communications. 7. Complete trust. 8. Cooperation vs. competition.

THEORY P

(continued from page 1)

people and reluctantly abdicate control to someone more technical.

The underlying premise of Theory P is: The more effectively we manage the people who program the computer, the better we can utilize the systems to support the information needs of the business. One directly influences the other. Just as the three theories of management are used to describe the corporate culture, Theory P represents a gauge for how a company values information and manages the resources needed to produce it. The theory, therefore, is an inherent part of an Information Resource Management (IRM) strategy.

The elements of Theory P were not derived form casual observations. Rather, they are based on fifty years of practical experience in the field, in a variety of industries. The principles contained herein were gained from actual programming assignments, managing programmers in both large and small shops, and consulting with literally hundreds of IT managers worldwide.

PERCEPTIONS

A particular management style is ultimately based on how a manager perceives an employee. For example, if a manager thinks a worker is lazy, the manager will spend more time supervising the individual. In contrast, if a manager has faith in the worker's judgement, the manager will allow the employee to supervise himself. Perceptions, therefore, plays a significant role in formulating a management style.

There are basically three perceptions management considers:

1. The worker's intelligence level - Whether the individual is considered capable of rising above their current position, or has exceeded their level of competency (the "in over their head" phenomenon). This is often gauged by the number of mistakes the worker makes and their ability to grasp new ideas.

2. The worker's motivation - Whether the worker is perceived as a self-starter and aggressively tackles assignments, or is lazy and needs to be coerced. This is primarily measured by the amount of time needed to supervise the individual.

3. The worker's attitude - Whether the worker is viewed as stimulated by their job and enjoys their work, or is adverse to work and apathetic to accomplishing anything. This can be analyzed by the amount of time spent conquering job assignments (obsessed with meeting a deadline versus a "clock watcher" mentality), and the employee's deportment as a professional (sharp and articulate versus slovenly).

Whether these perceptions are real or not, management will base their style of management on these variables. Many people understand the power of image, and often try to mislead others, particularly their superiors. Knowing these variables, many a worker has tried to convey a false image to their employer. For example, an impeccable taste in dress may be a charade for incompetence. Someone who spends an inordinate amount of time at the office, yet produces nothing, is not an effective measure of an individual's productivity. In other words, just because an employee is strong in one area, they may be weak in another. Management will ultimately base their opinions based on all three variables, not just one.

Before we consider how programmers fit within these three variables, let's define what I mean by the job title "Programmer." First, I deliberately avoided the term "Software Engineer" because this would imply the use of a scientific method to programming. Regardless of how one feels about the profession, this is hardly the case. Basically, the programmer's task is to convert human understandable specifications into machine understandable instructions. From this perspective, a programmer can best be characterized as a translator. Unfortunately, such a delineation chafes people in this profession. It is not the intention of this paper to insult or demean programmers, but rather to put their position into proper perspective.

I also avoided the terms "Systems Engineer" or "Systems Analyst" since such positions are aimed at total systems and not just the computer portions. A true "Systems Engineer/Analyst" studies business requirements, defines and develops business processes to implement the requirements, and specifies software requirements for programmers to implement. Regrettably, there are too few people performing this vital task and, consequently, the responsibility defaults to the programmer who is not necessarily equipped with the proper skills to perform the work. However, this is why the titles "Programmer/Analyst" or "Analyst/Programmer" are still dominant these days.

There are significant differences between a "Systems Engineer/Analyst" and a "Programmer." Whereas the former is a generalist who speaks the language of the

(continued on page 3)

THEORY P

(continued from page 2)

business, the latter is a detailist who must speak the language of the computer. This stems from the fact computers require precise instructions; the slightest syntactical or typographical error can cause it to fail, hence the need for someone who understands the nuances of programming languages. The two job functions require distinctly different skills and personalities. Combining the two functions is highly dubious and doesn't do justice to either.

INTELLIGENCE

Programmers tend to perceive themselves as free-spirited intellectuals who possess the magic of technology. Whereas the knowledge of the language is vital to performing their job, programmers often use it to bamboozle others and heighten their own self-importance. To outsiders, programmers are viewed as a sort of inner-circle of magicians who speak a rather cryptic language aimed at impressing others, as well as themselves. Such verbosity may actually mask some serious character flaws in their personality. Speaking in a foreign language may be amusing to a listener for awhile, but will inevitably alienate people over time.

It is not unusual for programmers to have problems socializing with others outside of their profession. Their language and technical interests tend to make them somewhat cliquish and the cause of peculiar sub-cultures within companies. It should come as no surprise, therefore, that management perceives programmers as nonconforming misfits who happen to hold the key to the corporate technology (e.g., the "Black Box" syndrome).

In reality there is only a handful of true programming geniuses in the field who are either independent contractors or are employed by computer hardware/software firms. There are few, if any, true programming geniuses in the average corporate shop. Regardless of the image they wish to project, the average programmer does not have a higher IQ than any other worker with a college degree. In fact, they may even be lower. Most exhibit little imagination and require considerable instruction and coaching in performing their job. When they have mastered a particular programming task, the source code becomes a part of their portfolio which they carry from one job to the next. So much so, that copying or stealing source code is actually the predominant mode of development in most companies. Consequently, there is little original source code being produced in today's software.

This copy/steal phenomenon also presents problems to companies who need to safeguard intellectual property. It is now too convenient for an employee to walk away with source code a company paid dearly for. Further, such action could seriously jeopardize a company's contractual agreements with hardware/software vendors or consultants.

To the programmer's credit, they usually possess a curiosity about technological developments. This must be carefully nurtured by management. Suppressing information may prevent the programmer from selecting a suitable technical approach, too much information may distract them from their job. An even balance must be struck. Programmers need freedom to explore technological alternatives without spending an inordinate amount of time in research.

There is also the problem that programmers tend to be somewhat faddish. It is not uncommon for them to recommend a solution that is technically fashionable, not necessarily what's practical. An elegant solution to the wrong problem solves nothing. It is important for programmers to learn to justify their technical recommendations from a business perspective. Failure to do so will inevitably result in a costly decision.

Because of the rapid speed technology changes, an ongoing curriculum of training should be developed to sharpen the programmer's skills. Further, a skills inventory should be developed to monitor programmer proficiency and to detect the need for additional training. The obvious concern here, from an employer's point of view, is that the programmer may leave the company after learning new skills, thereby squandering the company's investment in the person. Therefore, training should be on a 50/50 basis, the company should provide an ongoing curriculum either after-hours or on the weekends. It is important a company not spoon-feed an employee. If programmers do not demonstrate personal initiative to learn new subjects, the company should not waste time and money trying to teach them (see section on "Motivation" below).

It is well known that programmers generally abhor organization and discipline. Their desks are often littered with stacks of paper and other debris. The most common excuse is that *"a cluttered desk is a sign of a brilliant mind."* Nothing could be further from the truth. With little exception, sloppiness is a sign of mental laziness. In fact, many programmers deliberately appear disorganized to make it difficult to judge how they are progress-

(continued on page 4)

THEORY P

(continued from page 3)

ing on their work effort and reveal inadequacies in workmanship.

Mental laziness can also be found in planning and documenting software. Instead of carefully thinking through the logic of a program using graphics and text, most programmers prefer to dive into source code without much thinking; a sort of "leap before you look" philosophy. Inevitably, the only documentation describing the program is the source code itself, which has been written according to an individual's unique style and, of course, without comments and annotation. As a result, the software is difficult to modify and maintain. Eventually, the program is discarded and has to be re-written.

Key Observations:

1. Programmers exhibit an average intelligence level, no greater than any other professional with a college degree. They exhibit an average imagination.

2. Perhaps more than any other profession, programmers try to impress and intimidate others with their technical jargon. Such language usually masks inadequacies elsewhere.

3. Programmers are capable of learning new skills but must demonstrate a willingness to learn.

4. Without basic organization and discipline, programmers will become mentally lazy.

Recommendations for Management:

1. Be leery of programmers that are pseudo-intellectual. They are probably hiding something.

2. Improve communications within the programming staff by developing a standard glossary of terms. This will also be useful to outsiders who have to interface with programmers.

3. Carefully scrutinize technical proposals. Make the programmers justify it from a business perspective.

4. Adopt standards for documenting programs (e.g., graphics and/or text detailing the organization and logic of the program).

5. Develop standard development practices emphasizing quality and program re-usability. Demanding precision in development will result in superior performance.

6. Implement a skills inventory to monitor the talents and proficiencies of the programming staff. This is used to determine the need for additional training, as well to select the most suitable individual for a programming assignment.

7. Promote a program of on-going education, such as a training curriculum, the development of technical library, participation in professional associations, and technical certification programs.

8. Develop security measures to safeguard the company's intellectual property.

9. Recognize outstanding achievement even for the smallest of jobs.

10. Manage from the bottom-up. Delegate responsibility and hold people accountable for their actions. Teach employees to supervise themselves.

MOTIVATION

There are many motivational factors that influence people in performing their work: financial compensation, job security, dedication to their work, professional curiosity, and sometimes sheer ego. Management must be aware of these factors in order to properly stimulate people to tackle their assignments. After all, people will only work on those tasks they deem important. Unfortunately, not all programming assignments are glamorous. The vast majority of all corporate applications require some rather simple logic (e.g., read a file, sort data, write a report). It thereby becomes a challenge for management (and the programmer) to make a seemingly mundane task appear interesting.

ATTITUDE

The typical programmer has many insecurities and often laments he/she is being overworked, underpaid, and unappreciated. It should come as no small surprise that programmers feel more like journeymen with more of an allegiance to their profession as opposed to their company. As such, it is common for them to move from one job to another at the drop of a hat. Because they do not identify with their company, most are clock watchers and will only tackle those assignments they deem necessary and drag their heels on seemingly boring subjects.

Knowing this, management must effectively communicate to the programmer:

(continued on page 5)

THEORY P

(continued from page 4)

1. He/she is an important part of the corporation, but no more than any other worker.

2. He/she is being adequately compensated.

3. He/she is not being worked any harder than any other worker.

4. The importance of their work.

Like any other worker, the programmer needs to know they are leading a worthy life. Only with this sense of value will they lead a mutually beneficial life with their company.

CONCLUSION

Theory P is concerned with improving programmer productivity by addressing the reality of working with programmers as opposed to implementing the latest technological panacea. The more we understand how programmers think, the better we can manage them. Let us never forget they possess the same human frailties as the rest of us do. Sure, some have inflated egos, but most simply want a good basic standard of living and a little recognition for their work. In turn, the programmer must demonstrate responsibility and produce quality work. Bottom-line, there is no need to handle them gingerly, but as any other human resource.

Programmers fancy themselves as free-spirited individuals who resist discipline like a mustang resists the bit for the first time. But it must be done. A little discipline, organization and accountability can go a long way.

END

"PRIDE" Special Subject Bulletins can be found at the "PRIDE Methodologies for IRM Discussion Group" at:
http://groups.yahoo.com/group/mbapride/
You are welcome to join this group if you are so inclined.