**Managing the Software Enterprise is Different**

One of the significant aspects to being a good manager is to be able to motivate your employees. As we read in last week’s lecture there are several challenges unique to managing the software enterprise. Managing the software enterprise includes managing various IT personnel: including analysts, developers, technical specialists, and infrastructure support personnel.

Well performing IT professionals and project teams, specifically within the context of the criticality of these employees to contemporary organisations, can be described as critical success factor in the success of organisations. It is a mistake to assume that successful management and leadership techniques are universal, regardless of the profile and characteristics of those being managed or led. Managing virtual teams loosely united across time and space presents unique challenges, made even more challenging because many of these team members are IT professionals.

There are certain characteristics of IT professionals, the patterns of the attitudes of these professionals, their beliefs, and their behaviors that indicate managing them successfully may differ, and why. For example, we commonly accept that “geeks” have a passion for reason, an almost emotionless detachment. There is often a focus on seeking out problems and looking for solutions. These individuals possess a competitive spirit, and they often reject formal structural hierarchies. This makes it hard to be "the boss" and to get response from these professionals just because you are the boss. As a result, we find that “geeks” seek out informal group leaders when they are placed on project teams. This raises particular issues in the realm of managing virtual teams made up of IT professionals.

Additionally the work of IT professionals is different from the typical daily work of the average office worker. The nature of their work has multiple impacts. It affects those who choose to go into the profession. It affects those working in the profession, shaping them in various ways. Lastly, it affects those who are trying to manage or lead developers. IT professionals live in a work environment that is highly ambiguous, with multiple acceptable paths for solving nearly every problem. Their work is fraught with failure, and requests to try again. Their work requires deep concentration and focus. They often know more about their specific tasks and chores than their superiors do. They must be able to juggle multiple projects at the same time. They must understand the business as well as the technical aspects of what they do. All of this contributes to the uniqueness of managing and leading IT professionals.

We must also consider that the differences between software and more traditional goods and services have an impact on the performance of engineers. Software is an intangible product, traditional items are physical. Software can be easily duplicated leading to a possible failure of excludability. Traditional goods and services are not so easily duplicated. The cost of software goes into the production of an operational version and its evolution, not into its copying or duplication. The higher the number of users, the lower the cost per user.

Keeping these differences in mind, let’s now look at motivating IT professionals.

**Motivating Software Engineers**

Different people are motivated by different factors, and engineers are not always motivated by the same factors as their managers or by the same factors as the general public.

*Compared to the general population,* software engineers are much more motivated by possibility for growth, personal life, opportunity for technical supervision, and interpersonal relations with their peers. Developers are much less motivated by status, interpersonal relationships with subordinates, responsibility, and recognition (Boehm, 1984).

*Compared to their managers,* software engineers are somewhat more motivated by possibility for growth, personal life, and technical-supervision opportunity. Developers are much less motivated by responsibility, recognition, and interpersonal relationships with subordinates (Boehm, 1984).

The comparisons between engineers and managers are particularly interesting, and they help to explain some of the miscommunications that occur

between engineers and managers. If a manager attempts to motivate engineers the same way that he/she would like to be motivated, he/she is likely to fail.Engineers care little about the responsibility or recognition that would motivate managers. In order to motivate engineers, managers should emphasize technical challenges, autonomy, the chance to learn and use new skills, and career planning-and respect their personal lives.

The Keirsey Temperament Sorter (KTS-II) (Keirsey.com) is a personality instrument that helps individuals discover their personality type. According to the Keirsey theory, there are four basic temperament groups which describe human behavior: artisans, guardians, rationals and idealists. Keirsey has created tests for individual analysis, team analysis and organizational analysis. More detail on these tests along with a free “mini” personality analysis are available at [www.keirsey.com](http://www.keirsey.com).

Another source of insight into developer motivations comes from surveys conducted to determine the personality types of developers using the MyersBriggs Type Indicator (MBTI) test. There are 16 four-letter combinations, which means that there are 16 personality types. The MBTI measures people's preferences along four dimensions and comes up with a four-letter categorization (MBTI® Basics). The categories are:

Extroversion (E) or introversion (I)

Sensing (S) or intuitive (N)

Thinking (T) or feeling (F)

Judging (J) or perceiving (P).

Two studies found that computer professionals are much more "introverted" than the general population. "Introverts" in the context of MBTI doesn't mean quite the same thing it does in the non-MBTI world. In this case, it simply means that the person is more interested in the inner world of ideas than the external world of people and things. Somewhere between one-half to two-thirds of the computing population is introverted, compared with one-quarter to one-third of the general population (Lyons, 1985; Thomsett, 1990). The same surveys found that 80 percent of computer professionals have a preference for thinking (T) over feeling (F) compared to 50 percent of the general population. They have a preference for making decisions based on a logical and impersonal basis rather than on subjective personal values. This logical bent is reinforced by computer professionals' preference for judging (J) over perceiving (P); two-thirds of computer professionals are Js compared to about one-half of the general population. Js tend to live in a planned, orderly way, whereas Ps tend to be more flexible and adaptable.

The implication of this preference is clear: to appeal to a developer, it’s best to use logical arguments. For example, a lot of what's been written about motivation suggests that one key to exceptional productivity is to set seemingly impossible goals. That's fine for Fs, who might find such goals inspiring. But Ts will reject such goals out of hand for being "illogical." This is one reason that it is a rare group of engineers who will respond positively to an impossible schedule goal.

It’s best to keep in mind that different forms of motivation work with different people. Generalities about motivation can provide broad-brushed insights, but it’s important to try to identify the most effective motivation for each individual. Try to put yourself inside each team member's head and understand how he or she thinks.

As Frederick Herzberg points out, a kick in the pants doesn't produce motivation; it just produces movement (Herzberg, 2003). To tap into the best individual performance, it’s necessary to tap into developers' internal motivations by creating an environment in which they can satisfy their internal drives. When people are excited, they will put in long hours and enjoy it. To create this environment focus on the five factors that most influence software engineers motivation: achievement, possibility for growth, work itself, personal life, and technical-supervision opportunity(Boehm, 1984). " The following sections take up these five factors in more detail.

**Achievement**

One of the best ways to motivate developers is to provide an environment that makes it easy for them to focus on what they like doing most, which is developing software.

*Ownership*is one key to achievement motivation. People will work harder to achieve their own goals than to achieve someone else's. Microsoft's 105th employee, Chris Peters, points out that if you let developers create their own schedules, they take ownership of their schedules, and you get their buy-in (Cusumano and Selby, 1995).

*Goal setting* is another key to achievement motivation. Software engineers don’t respond well to objectives that change daily or that are collectively impossible to meet. Gerald Weinberg and Edward Schulman (1974 cited in McConnell 1996) conducted an experiment to investigate the effect of objectives on developer performance. They gave five teams of developers the assignment of working on five versions of the same program. They gave the same five objectives to each of the five teams; however, they told each team to maximize a different objective. They told one team to minimize memory required, another to produce the clearest possible output, another to make the most readable program, another to use the minimum number of statements, and the last group to complete the program in the least amount of time.

The results of this study showed that four of the five teams finished first in the objective they were told to optimize; one team finished second in its objective. They also gave each team a secondary objective, and, on those, three of the teams finished second in their secondary objective, one team finished first, and one team finished last. None of the teams did consistently well in all objectives.

The implication of this study is that developers do what they are asked to do. They have high achievement motivation: they will work to the specified objectives but they need to know what those objectives are.

Setting too many goals at once is a common problem. A review of 32 management teams found that, in response to the question, "What does the leader do that keeps the team from functioning more effectively?" The most common response was that the leader diluted the team's efforts with too many priorities (Larson and LaFasto, 1989). The researchers identified this as a major leadership blind spot because most team leaders gave themselves high ratings in the area of setting priorities. For best results, select one objective and make it clear that it is the most important one.

**Possibility for Growth**

One of the most exciting aspects of being a software engineer is working in a field that is constantly changing. An organisation can tap into an engineer’s motivation for growth by providing opportunities to grow on their projects. This requires aligning the growth goals of the organisation with the growth goals of the individual. Barry Boehm (1984) puts it this way: The principle of career progression indicates that it is in an organisation's best interest to help people determine how they wish to grow professionally, and to provide them with career development opportunities in those directions. This may seem like a somewhat obvious principle, but in practice there are many software organisations which follow strongly opposing principles.

An organisation can show interest in its developers' professional growth in any of these ways:

* By providing tuition reimbursement for professional-development
* classes
* By giving time off to attend classes or to study
* By providing reimbursement for purchase of professional books
* By assigning developers to projects that will expand their skill sets
* By assigning a mentor to each new developer (which shows both the mentor and the new developer that the organisation is dedicated to professional growth)
* By avoiding excessive schedule pressure (which tells developers that the real, top priority is getting the next product out the door regardless of the personal cost)

A focus on personal growth can have both short-term and long-term impacts on your organisation's productivity. Inthe short term, it will increase your team's motivation, causing them to work harder. Inthe long term, the organisation will improve its ability to attract and keep people from the top of the talent pool. (Naisbitt and Aburdene, 1985).

**The Work Itself**

Richard Hackman and Greg Oldham argue that, generally, people's internal motivation comes from three sources: They must experience meaning in their work; they must experience responsibility for the outcome of their work; and they must know the actual results of their work activities (Hackman and Oldham, 1980). Hackman and Oldham identified five dimensions of the work itself that contribute to these sources of motivation.

*Skill variety* is the degree to which your work requires you to exercise a variety of skills so that you can avoid boredom and fatigue. People find meaning in jobs that offer variety, even in work that is not very significant or important in any absolute sense.

*Task identity* is the degree to which your job requires you to complete a whole, identifiable piece of work. People care more about their work when they have a whole job to do and when they feel that their contribution matters.

*Task Significance* is the degree to which your work affects other people and contributes to social welfare. People need to feel that the final product has value. As Hackman and Oldham point out, people who tighten nuts on airplanes will feel that their work is more important and meaningful than people who tighten nuts on decorative mirrors. Likewise, developers who are allowed to meet customers and understand the big picture within which they do their work are likely to feel more motivated by their work than developers who are kept in the dark (Zawacki, 1993).

*Autonomy* is the degree to which you have control over the means and methods you use to perform your work-the sense of being your own boss, and the amount of elbow room you have. The more autonomy people have, the greater sense of personal responsibility they tend to feel for the outcome of their work.

*Job feedback* is the degree to which carrying out the job itself provides you with direct and clear information about how effective you are. (This is different from receiving feedback from a supervisor or coworker.) Software development provides great job feedback because the work itself-the program-provides the feedback: you can see immediately whether the program works when you run it.

One key to motivation is to control these five dimensions to create meaningful work and then match up that work with people who have a high desire to achieve. Robert Zawacki (1993) reported that his 15 years of research indicate that about 60 percent of a developer's motivation comes from the match-up between the job and the developer.

Another motivational aspect of the work itself is the degree to which the environment allows a developer to focus on the work itself instead of being concerned with related tasks such as finding supplies, fixing equipment rather than waiting days for a technician, filling out forms to procure extra materials. All this activity can divert attention from the work itself. Enforcing dress codes and strict work hours suggest that while the work is important, it is not the most important.

**Personal Life**

Although they are prioritized differently, achievement, possibility for growth, and work itself are in the top five motivators for both developers and managers. (). Those factors present a significant opportunity for developers and managers to understand what makes the other tick. Personal life is fourth for developers, fifteenth for managers.

Because of this disparity managers sometimes reward their best developers by assigning them to their highest-profile, highest-pressure projects. To the manager, the extra responsibility would be a treat, and the diminished personal life wouldn't matter much. To a developer, the extra responsibility is more trick than treat, and the diminished personal life is a keen loss. The developer interprets the manager's "reward" as punishment.

**Technical-Supervision Opportunity**

Managers are less motivated by opportunity for technical supervision than are software engineers. For an engineer, a technical-supervision opportunity represents an achievement. An opportunity to supervise technical work implies that the engineer has achieved a level of technical expertise sufficient to direct others. Technical-supervision opportunities are not limited to assigning one person to be the technical lead on a project. There are a number of different ways to use this as a motivator. The following are two examples of assigning technical supervision:

* Assign each person on a project to be the technical lead for a particular product area such as user-interface design, database operations, etc.
* Assign each person to be the technical lead for a particular process area such as technical reviews, integration, tool evaluation, etc.

**Rewards and Incentives**

In addition to the top five motivators, there are a few other factors that can affect software engineers’ motivations.

There are basically two categories of rewards: intrinsic and extrinsic. Intrinsic rewards produce non-quantifiable personal satisfaction such as a sense of accomplishment, personal control over one’s work and a feeling that one’s work is appreciated. Extrinsic rewards are external, tangible forms of recognition such as pay hikes, promotions, bonuses, and sales prizes. Both intrinsic and extrinsic rewards motivate behavior. (Motivation, 2006)

In*In Search of Excellence,* Peters and Waterman (1982) report that the companies that manage to stay in the top halves of their industries over 20-year periods make extensive use of *non-monetary incentives*. They concluded that nothing is more powerful than positive reinforcement. While everybody uses it, top performers use it extensively. As with any show of appreciation, it's the thought that counts. Be sure that these rewards say "appreciation" rather than "incentive" or "manipulation."

More information on intrinsic and extrinsic rewards can be found in Frederick Herzberg’s classic article “One More Time: How Do You Motivate Employees?” which was first published in 1968. Herzberg found that extrinsic incentives such as bigger paychecks and plush offices don’t necessarily make people work harder or better (Herzberg, 2003). When such motivators do succeed, the positive effects are short-lived. Pay matters in the sense that a company has a difficult time recruiting and keeping good employees without a competitive level of pay and benefits. Money can be a huge motivator, but it often motivates the wrong behaviors—for example, encouraging people to cut ethical corners to earn a bonus—and it does not build commitment.

*Other rewards and incentives* used by organisations include project t-shirts, special courses, sponsorship at conferences and special bonuses or end of project rewards for contributions to the project.

Another interesting incentive which comes from the "Hawthorne effect" could be termed *pilot projects.*Inone of the most famous experiments in motivation and productivity, Elton Mayo and his associates ran a series of tests on worker productivity from 1927 to 1932 at the Hawthorne Works of the Western Electric Company in Chicago (Hall & Fernandez-Ramil, 2007). Their aim was to determine the effect of lighting on productivity. First they tried turning the lights up, and productivity went up. Then they tried turning the lights down, and productivity went up again. They tried holding the lights constant, and productivity went up yet again (Boehm, 1984).

After many more experiments, Mayo concluded that increased productivity levels had nothing to do with lighting levels but were the result of conducting the experiments. In software engineering, it doesn't matter whether an improvement in productivity comes from new technology or from the Hawthorne effect. The implication for software projects is clear: run software projects as if they are experiments or pilot projects. To do this include some new methodology or new technology on each new project, and be sure that your team knows that this is a pilot project.

Motivation can be significantly increased or decreased by *performance reviews*. Andrew Grove (1982) indicates that a performance review is one of the most important forms of task related feedback that can be provided and that the review will influence a subordinate’s performance for a long time to come.

Performance appraisals are simply formal assessments of how well employees are doing their job. However, the impact of appraisals on compensation, motivation, training, career planning, and promotions is understandably significant. For example, the faculty-evaluation surveys that students complete at the end of a course are generally used to monitor performance and provide feedback. They can also be used to look for larger trends and satisfaction rates.

There are two types of errors or biases that can occur when evaluating an employee recency error and halo effect (Griffin, 2008, p. 388-389).

*Recency error:* Let's say that there's a saleswoman who has been average the entire year, yet booked a huge sale in the 4th quarter. This is fresh in her manager's mind, so he rates her as above average on her review, when in fact she is not. To modify this effect, managers should keep good records of employees' strengths and weaknesses, accomplishments and errors, throughout the year. These thorough notes can be used to base an evaluation, as well as concrete examples in case the employee gets defensive or disagrees with something.

*Halo error:* An example of this is when an employee that does excellent quality work but has a hard time meeting deadlines. Because this causes a lot of problems, you mark him as being a poor performer based on this single attribute. To moderate this effect the manager needs to make the effort to be impartial. It's natural to get along better with some people than with others, but an effective manager does not let this cloud their vision when looking at how employees meet the goals that have been established.

**Work environment**

In the 1960s, Fred Herzberg conducted research that identified two kinds of motivators. He differentiated between motivating factors (satisfiers), which stimulate performance when they are present, and hygiene factors (dissatisfiers), which degrade performance when they are absent.

Hygiene factors are the basic conditions a worker needs to work effectively. At best, hygiene factors create no dissatisfaction. At worst, their absence creates dissatisfaction. Adequate lighting is a hygiene factor because if adequate lighting is not present, the worker's ability to work effectively is impaired, and that hurts motivation. But adding extra lighting beyond a certain point does nothing to improve motivation. Good developers tend to gravitate toward organisations that provide work environments in which they can be productive-environments that meet their hygiene needs.

The following is a list of some of the hygiene factors which are important in the software enterprise:

Appropriate lighting, heating, and air-conditioning

Adequate desk and shelf space

Enough quiet to allow concentration (including the ability to turn off the telephone)

Enough privacy to prevent unwanted interruptions

Access to office equipment and supplies (such as a photocopy machine and a fax machine)

Unrestricted access to a computer

Up-to-date computing equipment

Immediate or near-immediate repair of broken computer equipment

Up-to-date communications support (including email, individual telephones, voice mail, and ready access to well-equipped conference rooms)

Applicable software tools (word processor, design tools, programmer's editor, compiler, code libraries, debugging aids, and so on)

Applicable hardware (for example, a color printer if you are working on a graphics application and expect most of your customers to print in color)

Applicable reference manuals and trade publications

Auxiliary reference books and on-line reference aids

At least minimal training in new computer software, tools, and methodologies (additional training can be a growth motivator)

Legal copies of all software used

Freedom to set work hours-both generally (choosing to work 8-5, 11-8, or some other hours) and specifically (taking the afternoon off to attend a child's school play)

**De-Motivators**

Just as hygiene factors can motivate engineers, not meeting a hygiene factor adequately can lower motivation and morale. In addition to this, management can damage motivation and morale in other ways.

*Management manipulation***.** Developers are sensitive to being manipulated by management. Developers tend to deal with issues head-on, and they want management to deal with developers in a straightforward manner.

A few managers try to manipulate developers by setting phony deadlines, and most developers can smell a phony deadline 100 yards away. It’s important that when engineers question deadlines or feature creep, answers don’t appear to be evasive and manipulative.

*Excessive schedule pressure.*If the deadline is real, it still might not be realistic. One of the quickest ways to drop motivation to zero is to present developers with an impossible deadline (Boehm, 1984). Few people will work hard to meet a deadline they know is impossible, especially if they are MBTI type Ts (those who respond to logic more than to emotion).

*Lack of appreciation for development's efforts.*Ken Whitaker (1994) described a meeting his software-development team had with his organisation's marketing branch at which the marketing representative indicated that the developers were sandbagging the product. Whitaker believes that this is a common dynamic-people don't *see* the work developers are doing, so they don't think they're doing much and therefore must be sandbagging. Developers are extremely self-motivated, and find being accused of "loafing" to be very de-motivating.

*Inappropriate involvement of technically inept management.*Developers can be motivated by technically inept managers as long as those managers recognize that they are technically inept and confine their control of the project to non-technical decisions. If they try to meddle in technical decisions that they don't understand, they will become the butt of the development team's jokes, and most people cannot be motivated by someone they do not respect.

*Not involving software engineers in decisions that affect them*creates the impression that the manager either doesn't care about the development team or that management doesn't respect them enough to want their contributions. Here are some instances in which management must involve engineers if it wants to keep their motivation high:

Committing to new schedules

Committing to new features or feature modifications

Hiring new team members

Volunteering developers for short -term assignments outside the project

Designing the product

Making technical trade-off decisions

Changing office space

Changing computer hardware

Changing software tools

Committing to deliver work products that might or might not already be planned by the team such as a prototype for use by the customer or a prerelease version of the product.