# SOFTWARE ENGINEERING MANAGEMENT FOR PRODUCTIVITY AND QUALITY

K. S. White, Jefferson Lab, Newport News, VA, USA

*Abstract*

Since the advent of electronic computers, people have struggled to develop effective software engineering processes. While these processes are similar to those used by hardware engineers, the software industry has earned a reputation for late delivery of inadequate products. Most software managers are looking for ways to deliver quality products faster, or with fewer resources. The development time and product outcome of any software project can be influenced by four variables: the product characteristics, the people involved, the processes they use, and the underlying technology. In order to have an impact on the productivity of a software development effort, the manager must focus on and balance these areas. This paper will discuss effective ways to improve productivity by using this approach.

## 1 INTRODUCTION

### *1.1Background*

In the late 1940's, as the first electronic computers began to appear, the software engineering industry was born. Since that time, project managers have been experimenting with ways to manage software development. Over fifty years later, a process that began with fairly small, single purpose machine language programs developed by an individual engineer or mathematician, has grown into mammoth, multiyear projects staffed by teams of software engineers. Early projects typically ran in batch mode on a single platform, whereas today's projects must run interactively on multiple hardware platforms and operating system configurations. Software development has ballooned into a multibillion dollar industry involved in virtually every new government and private sector project. A few low level languages have been superceded with numerous high level languages, often complete with tools to aid in development. However, the benefits gained from the relative ease of programming with high level languages and tools have been overshadowed by the increased complexity of software projects.

Most software project managers are constantly looking for ways to deliver better products faster. On the surface, a software development project is a lot like any other project. There are schedules to be worked out, resources to balance and project goals to achieve. However, a big difference comes from the fact that each new software project is unique. Programmers are fond of modifying existing code, but with large, complex projects, there are many pieces that must be created and integrated and code reusability is often limited. Software sharing, the practice of writing commonly needed code so that it can be used for a variety of projects, is an increasingly popular concept, but cannot always be used in practice due to unique and complex project requirements.

For a software project to be a success, it must create a product that is what the customer wants, and the product must be delivered within a reasonable time frame. Faced with limited resources and unlimited work, the productivity of the software team becomes critical to project success. Productivity is a measure of the efficiency with which resources are employed to produce goods or services, in this case software. As managers approach software development projects, knowing how to facilitate productivity can be very helpful.

### *1.2 Common Problems*

With software development, one problem occurs when projects begin without adequate product requirements. Another problem involves the imprecise practice of generating time estimates for software projects. Starting with a poor time estimate can doom even the most efficient development team to late delivery. Another factor working against success stems from the very nature of software. There are as many ways to design and program as there are programmers. There are also many opportunities to introduce errors. Add this to the fact that modern programs are typically too large to exhaustively test and many potential problems loom before the project even begins. Couple these problems with increasing pressure to produce more with fewer resources and the software project manager's job becomes a real challenge.

## 2 PRODUCTIVITY FACTORS

### *2.1 Factors*

While there have been volumes written on the topic of software development and many methods put forth to increase productivity, there are four factors that are recurring themes [1][2][3]. Effectively managing these four areas can help a project manager maximize productivity. The outcome, including the efficiency of development, of any software project is heavily influenced by the people involved, the processes they use, the product size and characteristics, and the underlying technology used to implement the product. In order to have an impact on the overall productivity of a software engineering effort, the manager must focus on the characteristics of the product, the along with the combination of people, processes and technology used to

develop the product. It is not enough to do well in one or two of these areas; all four must be considered and balanced when there is a desire to raise productivity. Within each area, there are many ways to improve.

## 2.1 Products

The product is the desired outcome of a development effort. The characteristics of a product, as defined by the requirements, affect the overall development time. At this phase of a project, the manager's primary concerns center on the scope and the quality of the requirements.

The requirements document should cover all functions the product must perform, and therefore is the best indicator of the scope and size of the project. The number of functions needed and the complexity of these functions directly impact the size of the product and therefore the time it will take for each stage of development. Working with the customer in the early stages of a project, as the requirements are defined, the project manager can influence the product definition and therefore the development time. For example, if the requirements call for a large product with many features, the manager can suggest to the customers that the most critical features be called out explicitly so the work can proceed in a phased manner. This will enable the programmers to design the product with the full requirements in mind, but deliver a partial product sooner, satisfying the customers most urgent needs. The manager can also suggest acceptable, less complex, alternatives to requirements that may be very onerous to implement. He can also make the customer aware of the direct relationship between the scope of the requirements and the time to delivery. Customers then may be less prone to "gold plate" the requirements if they understand how such extra, unnecessary features will slow the process.

Aside from the scope of the product requirements, one must also ensure the quality of the requirements. The quality or lack thereof, of the requirements is usually mirrored in the final product. Poorly written requirements can effectively extend the project duration as customers file numerous change orders, requesting work to turn the requested product into the desired product. This issue goes beyond the written document, which must be clear and complete, and extends to the understanding of the requirements. Often, product requirements are not well communicated and understood by the customers or the programmers. This situation leads to wasted time as development proceeds down the wrong path, with programmers creating a different product than the customer wants. An effective way to prevent this is tasking customers and programmers to develop the requirements together. During this phase, the customers and the developers need to document and understand a common vision of what the product will be. This may involve developers learning some of the language of the customer's business, and the customers learning some software terms. Having some common language helps the programmers and customers communicate better throughout the project, a key to good results. Often, the requirements stage will become easier once customers and developers have worked together this way a few times.

Beyond the requirements phase, continuing to include the customer in the development process helps ensure the developing product stays on target to meet the product goals. One common way this can be accomplished is to have the developers do some rapid prototyping of the product framework and interface, then allow the customer to experience the evolving look and feel of the product. Customer feedback at this stage can save a lot of programming changes later in the process and helps to further define the product. The more closely programmers and customers work throughout all phases of development, the better the product will meet the customer's expectations. This improves quality by preventing the product from turning into something different from what the customer wants, thereby saving much time in the end.

## 2.2 People

It is important to have the right people working on a software project. This means software engineers who are knowledgeable, skilled and satisfied with their work. It is even more important to have a project manager that works well with people and values their contributions. No single person can ensure project failure more effectively than the project manager. Studies indicate that people related issues have the biggest impact on programmer productivity and software quality [1]. Individual productivity has been reported to vary by up to a factor of ten between programmers with similar experience levels. Team productivity has been reported to vary by as much as a factor of five. With people issues having so much potential power over the outcome of a project, it makes sense for leaders to learn what they can do to have a positive influence in this area. The most common people issues are staffing, motivation, and the work environment.

When selecting staff, it is important to take the time to find the best possible people. The selection process should not be limited to the candidates' professional skills and knowledge. Rather, it should provide ways to determine what type of employee the candidate is likely to be, whether the candidate fits with the work environment, and if he has the potential to contribute to the team framework. Since many software projects are now large enough to require a team of people, one should pay attention to the composition of each team in terms of developers skills, task assignments and compatibility. It often takes time for people to learn to work together as a team, and some managers have success by providing some team building training or activities to jump start this process. Another key to having good staff is providing

training. Ensure developers have the time and opportunity for training to develop the project skills they need.

Having selected excellent staff, or making the best of existing staff, motivation is necessary to build productivity. Motivated individuals tend to work longer and harder than others. Different factors motivate different people. Part of the project manager's job is to know team members well enough to know what motivates them. Among the top motivators for programmers are the opportunity for professional growth, achievement, and challenging work. Some people measure their worth in terms of money, and are motivated by a bonus or raise as a reward for a job well done. Others enjoy the opportunity to travel to professional conferences and the recognition that comes from presenting their work among peers. Some people want their work to be recognized within the organization. This can be accomplished by giving awards, or just taking the time to stop by and comment on the progress of their project. Morale building activities, such as getting the group together for a social meal or activity, can also be effective motivators as well as team builders. In any event, knowing what motivates your team, and doing your best to provide such opportunities can improve productivity for any project.

Many organizations pay little attention to the work environment. To facilitate productive programming, the workplace should provide quiet spaces with a minimum of interruptions [3]. Programming is the type of work that requires periods of concentration, and each time the thought process is broken, it takes some time for the developer to become reimmersed in thought. Minimizing interruptions due to telephones ringing, pagers buzzing and people dropping by can help programmers make more effective use of their time. For this reason, ideally, there should be no more than two programmers per office. If people are to be paired together in offices, be sure to pay attention to putting compatible people together to avoid unnecessary stress. To enhance teamwork, all team members should be located in close proximity. Programmers also need physical resources such as computers and test equipment, and the workspace should provide adequate space for these items. There should also be a place, apart from the offices, where the team can meet informally to discuss the project.

### *2. 3 Processes*

The processes involved in software development are put in place to standardize development practices, a very necessary element to ensure quality. Good processes will also result in consistent documentation. A full range of recommended practices for various levels of product quality has been developed by the Software Engineering Institute and is called the Capability Maturity Model [4]. Utilizing such resources as a model can benefit any type of serious software effort.

While studies over the past decade support the importance of good processes as a way to reduce delivery time, it is important that these processes are value added. This means the process must make development easier or better than it is without the process. Because processes are sometimes applied in a strictly arbitrary bureaucratic fashion, many programmers have come to dread the very word. The best way to insure a process does not fall into this category is to allow the people who will use the process to develop it. In order to make this approach work, a manager must first think carefully about the motivation for implementing or changing the process. The manager must then clearly communicate the goal to the programmers and have them develop the process. This will result in a process that the programmers will use, because they developed it to make their work easier and better.

### *2. 4 Technology*

The technology chosen for a project has the potential to shorten or lengthen the development time. When selecting a technology, it is important to consider the product, the stability and longevity of the supplying company, and the experience of your staff working with similar products. It is often possible to gain productivity by switching to a newer technology that is easier to use. For example, in the past, changes from machine languages to high level languages have facilitated a tremendous productivity improvement during the coding phase of a project. One must be sure to plan for the inevitable time needed to make the switch, including purchasing, installing and testing new products and staff training time. It is important to be aware that selecting a very new, unproven technology can add substantial time and risk to a development project. This time is typically spent working out problems with the new technology instead of developing the product.

## 3 CONCLUSIONS

By increasing their awareness of productivity factors, and using this information when managing software development projects, managers can have a positive impact on productivity. Many of these same techniques also help to improve product quality and customer satisfaction.

## REFERENCES


[1] S. McConnell, "Rapid Development", Microsoft Press, 1996.

[2] D. Phillips, "The Software Project Managers Handbook", IEEE, 1998.

[3] T. DeMarco, T. Lister, "Peopleware", Dorset House, 1987.

[4] Carnegie Mellon University/Software Engineering Institute. "The Capability Maturity Model", Addison-Wesley, 1995.