A Practical Approach to Predictable Software Development Performance in Small to Medium Size Software Development Organizations

Steven Teleki
Vice President, Software Engineering
Y&L Consulting, Inc.
teleki@computer.org

***Abstract*- Every manager should be concerned about economic performance. Economic performance hinges on technical (product or service) performance. In case of many organizations, success depends on having the necessary software to operate, provide services to customers, or to design and develop new products. Most of this software is purchased or contracted out by the organization, but at least a critically small portion of the software that the organization needs for its success has to be developed by software folks that are part of the organization. Even though software is developed by teams, it still is an *individual* activity. Thus to get predictably high software development team performance, it is imperative that software development managers understand how to achieve predictably high individual software development performance.**

I. Introduction

A predictable performance enables an organization to plan and execute with confidence its mission. The strategy to achieve predictable performance in software development must consider the entire chain of activities leading up to developing software, from concept through the actual writing of the code, all the way to maintenance. This article will focus on what is Software Development Performance in general what is performance at the individual level, and describes methods for managers to grow the capabilities of the folks working in their organization.

II. Software Development Performance

What do you know to be *important*, but you are *unable* to measure? Way back on October 22, 1707 it was longitude (or how far east or west you have traveled) [1]. When Sir Admiral Clowdisley Shovell commandeered four warships onto the rocks at a tail of islands at the southwest tip of England on that fatal day, 2,000 lives were lost as a result of the admiral's inability to measure longitude.

For software managers today measuring software development performance would be as important as measuring longitude was in 1707 for a British Navy admiral. Yet today, we are still struggling with this notion just as much as the admiral struggled with longitude in 1707.

A. What is Software Development Performance?

Software Development Performance is the measure of an organization's capacity to fulfill its software development obligations. It includes all activities that contribute to the creation of software systems. Thus this measure not only refers to the activity we call "programming" or "coding," but rather it encompasses all activities along the entire economic value chain of software creation, from the concept phase through maintenance.

Software Development Performance must encompass all activities, since software is no longer created in isolation (if it ever was), but rather software is created together with other significant parts of complex systems.

This means that the organization's Software Development Performance must include the performance measures of the activities that are part of the value creation chain, such as:

- Identifying problems
- Understanding the user's tasks and work
- Modeling the user's environment
- Capturing, analyzing, and modeling requirements
- Designing and architecting software
- Implementing software
- Testing, verifying, and validating software
- Understanding and deploying new technology
- Supporting and maintaining complex system, etc. …

B. Why Software Development Performance Matters?

It is very common for an organization's economic performance to depend on the organization's technical performance. Since almost every product or service contains a significant software component, the organization's Software Development Performance greatly influences the organization's economic performance. In most of today's businesses it is nearly impossible to create and launch a new product or service on time and on budget if the organization's software development performance is poor.

Imagine that you are trying to provide a new service to your customers, but the software that is supposed to handle the new pricing is not yet complete. You cannot launch the service until the software is finished. Or, you want to market a new device, but the software that can configure the device, has not finished its test cycle, because the team found too many defects in it. In these cases, even though everything else might be ready, software will keep you from successfully moving forward with your business plans.

As a result, we look at Software Development Performance because it is a key success factor in determining the short-term and long-term success of an organization. As managers, our goal is to understand the organization's Software Development Performance and then influence the factors that control the Software Development Performance.

III. Individual Software Development Performance

Software development is knowledge intensive work. All work is performed by people whom Drucker calls "knowledge workers" [2]. There are two key managerial tasks related to managing knowledge workers: identify the talents and non-talents of the people and continually develop people and grow their capabilities.

A. Identify Talents

The goal of predictable organization or team performance can be achieved if it is built upon predictable individual performance. Identifying the talents of an individual can

constitute the first step toward helping the person achieve predictable individual performance.

Buckingham and Coffman suggest in [3] that "Every role performed at excellence requires talent." There is an important lessons for managers in the previous statement: If you want excellence, you must know what the talents are; there is just no way around it.

The manager's task is to work with each individual to discover where his or her talents are and then cast the individual in a role that requires the talent that the individual has. Buckingham and Coffman give a definition of talent that can serve as a starting point for the talent discovery process: "A talent is a recurring pattern of thought, feeling, or behavior that can be productively applied" [3]. By this definition, talent is not something special that only a chosen few possess, but rather is a natural occurrence that all of us have.

According to the authors of [3] we were not educated to think about talent, or to even recognize one when we see it outside of sports or entertainment. Even when we are asked, we have difficulty to name what our talents are and we keep talking about the things we do or what our job or role is.

B. Cope with Non-Talents

A person has many more non-talents then talents. The manager needs to compensate for a person's non-talents by building teams of people with complimentary talents and by working with the individual to devise mechanisms to compensate for the non-talents.

In monthly or bi-monthly one-on-one conversations between the manager and each of his or her team members the focus should be mostly figuring out how to improve the person's talents (70%) and a bit on how to minimize the effect of the non-talents (30%). There is no use in trying to make somebody an expert in areas where they have no talent. This will only lead to frustration on everybody's part.

Identifying the talents and non-talents of individuals in small organizations is vital, since in a small organization each person carries a larger percentage of the overall workload, and thus any improvement in the person's performance can significantly impact the organization's overall performance.

IV. Methods for Growing Capabilities

Drucker points out in [2] that knowledge businesses may have a better option for considering the knowledge workers as assets, rather then costs on their balance sheet. Assets are meant to grow and to produce more, but costs are meant to be cut. What does it mean to grow people's capabilities? It means that people continually improve their capacity to create by learning, practice, and doing.

The farmers of the world had figured out millennia ago: if one eats all one's seeds during the winter and fall, then there will be nothing to put in the ground in the spring. As a result there will be nothing to harvest the next fall. Saving the seeds in the software business means enabling the folks in the organization to invest around 10% of their time into learning and professional growth related activities. Where will next year's performance come from if not from the investments made today?

Some of the current learning practices can be very expensive: conferences, training courses, tutorials, consultants, etc. They are important, but one conference a year will not provide sufficient learning opportunity that is needed to grow a person professionally.

For managers on a limited budget there are plenty of low-cost options: organize study groups, and lunch & learn, sign up people for professional associations offering regular presentations, open source projects, assisting people on technical mailing lists.

A. Study Groups

For $50 (the price of a book) you can buy a person's commitment to learning for an 8 to 10 week period. Study groups work best when at least some of the participating folks are working in the same group or unit.

B. Lunch & Learn

This is one of the most productive forms of continuing education for knowledge workers. The advantage of the format comes from having a member of the organization present on a topic of either their or the group's choosing. When people prepare to deliver a presentation to their peers they learn a tremendous amount.

C. Professional Associations

Encourage the people from your organization to attend local user group presentations and to become involved in trade associations. All of them provide ample low-cost learning opportunities. Most of these groups have monthly meetings with a presentation on a technical topic. If you're reading this paper, chances are you already belong to the IEEE. See that the people working for you join the organization as well, maybe make membership in IEEE a company benefit.

D. Open Source Projects

Encouraging people to contribute to Open Source projects allows them to see how would work be when there is a very different set of constraints placed on them. Having literally potentially millions of people look at your code can be troubling for some, and exhilarating for others. The opportunity is to learn to live in a glass house and work in an environment where you have to earn the respect of your peers without a formal hierarchy.

E. Technical Mailing Lists Participation

Participating as a contributor on a technical mailing list can be both gratifying and challenging at the same time. When an individual responds to a question, then others might question the validity of the response, and make corrections to it. This can hurt at first, but as the person acquires more knowledge, the answers can become the gold standard of the domain where he or she is contributing.

Conclusion

While it is impossible to describe a complete system for predictable software development in a short paper, the goal here was to give the reader a brief overview of the personal and organizational software development performance challenge.

Acknowledgment

Thanks to Dan Massey for his constructive feedback on the early draft of this paper.

References

[1] D. Sobel, *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time*. New York, NY: Penguin Books, 1995.

[2] P. Drucker, *Management Challenges for the 21st Century*. New York, NY: HarperBusiness, 1999.
[3] M. Buckingham, C. Coffman, *First, Break All The Rules.* New York, NY: Simon & Schuster, 1999.

Revised: 15 July, 2004