1

Software Programmer
The Management Perspective

G P Sudhakar

The aspects, objectives setting for the programmer, training the programmer, evaluating performance, evaluating technical skills, evaluating soft skills, training needs, career paths, from technical expert to Project Manager, moving the programmer across teams, are discussed in this article. It also discusses the different ways of evaluating the performance of software engineers in an organization. Some of the measures to evaluate the performance such as LOC (Lines of Code) per man month, function points, customer satisfaction, number of defects found, number of defects fixed, etc., are discussed. SEI-CMMI, and PCMM are also discussed.

Introduction

When managing programmers, every project manager thinks about how to measure the performance of these programmers. Are there any benchmarks for doing that? How do other companies in the industry measure the performance of software engineers? What kind of parameters can be considered while evaluating

the performance of software engineers? Answers to all these questions can be found in this article.

When a software engineer is recruited newly into the company, he/she undergoes the induction into the organization and into the project. Once an organizational induction is over, the engineer is assigned to a specific project to work as a team member or in a specific role in the project. The project manager introduces the new team member to all the other team members in the project. Then the new engineer undergoes the project induction and hands on training from the senior members of the team. As part of the formal process, the project manager sets the objectives to the new team member.

Setting Objectives

The objectives setting can be done formally or informally. In majority of the IT organizations, it is formal. The project manager sits with the new team member and sets the objectives for the coming one year or for the coming appraisal period. These objectives include the individual targets that the team members should achieve in the forthcoming year. Usually, these objectives will be inline with the organizational objectives and the team objectives. They include the technical and non-technical objectives, which the team members should achieve in the coming appraisal period.

The technical objectives include the modules or the tasks that the team member has to execute in the project or the quality, customer satisfaction levels that the team member should reach, etc. The non-technical objectives include the communication, leadership, and presentation skills as well as specific targets. This objectives setting process takes place when a new member joins the team. The respective project manager documents these objectives and communicates them to the human resource department. A copy of the same will be with the team member as well.

Providing Training

The new team members will undergo training in the organization and in the project. Those trainings can be termed as on-the-job training and hands on training. The senior programmers or the designers in the team will train the new team

members in the project specific technologies and domains. Usually every new engineer goes through the project documentation and the training manuals of the project. The project manager makes sure that every new team member is properly trained on the relevant technologies and modules. Usually, the work assigned to the team member will be inline with the objectives set by the project manager.

Evaluating Performance

Once the new team member completes one year or one appraisal period, it is time to evaluate the performance of the engineer. The project manager will conduct a formal performance evaluation under the guidelines given by the HR department. The project manager evaluates the software engineer's technical and non-technical skills.

The non-technical skills such as communication, team play, leadership skills of the engineers are evaluated during the appraisal time. Other things like business etiquettes, ethical behavior and mingling with organizational culture are also evaluated for the engineer during performance appraisal period.

According to Tom (1982), taking a poor performer out of the team may be more productive than adding a good performer to the team. The reasons for poor performance of programmer include lack of job satisfaction, lack of identification, little belief that professional behavior will result in rewards, lack of sense of professionalism at the programmer's end (Robert, 1990).

According to I M Wright (2007), it is better to take the end result into consideration, while measuring the performance of software engineers rather than consider the intermediate results. Some organizations even take the feedback from the customers while evaluating performance of software engineers.

Evaluating Technical Skills

The technical skills such as designing, coding and testing of the software engineers are being evaluated during the performance appraisal time. There are many ways to measure the technical performance of software engineers. They include Lines of Code (LOC) per man month, number of Function Points (FP) implemented in a man month, time taken to implement a function point, customer satisfaction,

number of bugs found for kilo lines of code, number of bugs fixed, time taken to fix a bug, etc.

Measuring Lines Of Code (LOC) per Man Month

This is one way of measuring the performance of software engineer. However, this may not be the correct measure to compare the performance of software engineers working in a team. It is not appropriate to say that the programmer who has written 10,000 Lines of Code is more productive than the programmer who has written 1,000 Lines of Code because the code of the later programmer may be very complex to compare with the other. Or the programmers might have used two different programming languages. If the programming languages are different, it cannot be measured by the number of lines of code. It is not known that the programmer who has written more number of Lines of Code might have written code, which may not be necessary. Hence, this may not be considered as an exact measure to evaluate the performance of software engineers.

Function Points Measure

Another way of measuring the performance of programmers is to measure the number of Function Points he/she has implemented in a man month time or the time taken to implement single Function Point. Because these function points are programming language independent, it can give the reasonable measure of programmer's performance. To use this measure, there should be an expert Function Point estimator in the project team to estimate the function points accurately. In modern day organizations, separate project function point estimators, do this activity.

Complexity Measure

Some of the organizations even measure the complexity of the code written by the individual programmer. This is useful to find out who has written the complex code in the project. Usually, the order of complexity of the algorithm can be found out by using different algorithm analysis and design techniques. It is best practice to measure the individual programmer's performance by using these techniques.

Combination of Measures

As we know the Lines of Code (LOC) per man month measure cannot give the correct measure to evaluate the performance of programmer, it is best practice to have a combination of all these measures while evaluating the performance of software engineers. The other measures include number of use cases implemented in man month, number of bugs found per kilo Lines of Code, number of bugs fixed, time taken to fix a bug, and backward traceability etc. Many organizations in the industry use the combination of all these measures while evaluating the performance of software engineers.

According to Suresh (2008), defects, injected during the design, need more time to fix than the defects injected during the coding time. Software Engineering Institute (SEI) of Carnegie Mellon University (CMU) has defined a Personal Software Process (PSP) to evaluate the individual programmers performance. SEI has also defined a Team Software Process (TSP) to evaluate the performance of teams as well. The Personal Software Process takes into consideration of the individual programmer and the time he/she has taken in each stage such as design, coding, and defect correction etc. According to Suresh (2008), some of the metrics of PSP take into consideration of planned vs actual estimates of tasks of the individual programmer.

Evaluating Soft Skills

During performance appraisal, the non-technical skills of the programmer such as communication skills, comfortability with team, meeting the organizational objectives, aligning with organizational culture, meeting the organizational code of conduct, behavioral aspects of the programmer are also evaluated. These aspects will have significant weightage in the programmer's performance appraisal process.

Usually, the organizations are observing the engineers behavior in teams because the teams rather than individuals are executing many of the projects in organizations.

Leadership Training for Future Roles

Based on the performance appraisal of the programmer, training needs are identified for the individual programmer. Based on the interests of the engineer and project requirements, he or she will be given advanced technical training or

managerial training. The programmers who took the advanced technical training will become the future designers and architects and the programmers who took the managerial training will fill the future project management roles in the organization.

Here, one important thing to be kept in mind is that the best programmer may not become a best manager. Because, the project manager should need entirely different skill set which include decision-making, negotiation, problem solving, leadership, conflict resolution, quantitative, conceptual and communication skills.

Career Paths for Programmers

The programmers can take the path of either management or Architecture/Design path. Based on the programmer's interests, he or she can get trained towards that direction. Usually, experienced programmers become designers or technical leads for the product. Those who are interested in communication and team leading roles may become the project leaders and project managers in the organization.

If a programmer chooses the managerial path, he can grow to project, program, portfolio manager and Director (Engineering) and VP positions in the organization. If he or she chooses the technical path, they can become Technical Leader, Designer, Architect and Chief Architect for the product.

According to John (2006), the programmers can pursue senior technical positions, business and system analysis positions, project management positions and management roles. Making the shift to management can be done on the job by taking more responsibility, polishing problem-solving skills and using creativity at workplace (John, 2006).

From Technical Expert to Project Manager

According to Alfonso (2007), some companies assume that the next position to programmer is project manager without even taking the consent and interest of the programmer into account. Project management roles need special skills other than programming skills. Those skills include communication, presentation, leadership, problem solving, decision-making, negotiation, and conceptual skills.

A programmer can become project manager by taking more responsibility and leading other team members in the team. He needs to have an understanding

of project scope, time, cost, quality, human resources, communications, risk and procurement management techniques and tools. He needs to have the knowledge of the project management knowledge areas mentioned above. One should gain skills and experience in managing projects and people to grow in the managerial paths.

Moving Across Projects

Once the programmer spends sometime in one project, he/she will be moved to another project, based on the project completion, or needs of other important projects within the organization. He/she can move across the domains or across the technologies or across the verticals.

Usually the senior project managers, program and resource managers move the team members across the projects based on the organizational requirements and the team member's needs or interests.

Promotions/Transfers/Onsite

Based on the programmers' performance appraisal, promotions, demotions and transfers take place in the organization. The promotions can be towards more technical roles or towards managerial roles. If the organization feels that the programmer's performance is not meeting the requirements of the project, they can even move him to the low level technical roles in other projects or even some disciplinary action can be taken.

There are some industry standards evolved in measuring the organizational performance and individual performance. Capability Maturity Model Integration (CMMI) and People Capability Maturity Model (PCMM) come under those standards used to evaluate the maturity of the processes in the organization. These models were developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. Explanation of these two models follows.

CMMI (Capability Maturity Model Integration)

The Software Engineering Institute (SEI) at Carnegie Mellon University issues the CMMI standards. Capability Maturity Model Integration is a process improvement approach, which provides organizations with essential elements of

effective processes. It can be applied to projects, departments or to the entire organization. CMMI helps in integrating the organizational functions with the quality processes and gives guidance as well as sets goals for quality processes.

One of the benefits of CMMI is that more linking is possible between engineering activities and organizational business objectives. It also fully complies with relevant ISO standards. The different levels of CMMI include initial, managed, defined, quantitatively managed, and optimizing. CMMI is being adopted across Americas, Europe, Asia, Australia and Africa.

PCMM (People Capability Maturity Model)

People Capability Maturity Model (PCMM) is also from SEI. It focuses on continuous improvement and development of human aspects of the organization. There are different maturity levels defined in PCMM, which deal with the human resources, and organization development related aspects. PCMM gives a framework, which helps the organizations in addressing the critical people issues.

Conclusion

More research is needed in order to find out the benchmarks for evaluating the performance of the software engineers. Majority of the modern organizations use the combination of different measures mentioned in this article. There is scope for the development of standards in measuring the performance of programmers. As a team player, both the technical and soft skills of the programmer together make him/her a good contributor to the project in turn to the success of the organization. Hence, there should be good and reasonable measures to evaluate the performance of software engineers working in the industry. With the application of PCMM frameworks, the organizations can handle the people issues in a more matured way.

(G P Sudhakar, MCA, M.Tech, EMBA, PMP, MIMA, ADM (PhD), Consulting Editor, Icfai Research Center, The Icfai University, Hyderabad. He can be reached at sudhakar@iupindia.org).

References

1. Alfonso Bucero (2007), “From Technical Expert to Project Manager”, *PM World Today*, Vol. IX, Issue VIII, August 2007.

2. DeMarco Tom (1982), *Controlling Software Projects: Management, Measurement & Estimation*, Yourdon Press, New York, 1982.
3. Donald J. Reifer (2004), "Industry Software Cost, Quality, and Productivity Benchmarks", Available at *http://www.compaid.com/caiinternet/ezine/Reifer-Benchmarks.pdf*, April 2004 (Accessed on 17-Nov-2008).
4. E. Naveen Kumar (2006), "Seven Commendments of Software Engineering", *Icfai Journal of Information Technology*, June 2006.
5. G. Gordon Schulmeyer, "The Net Negative Producing Programmer", Available online at *http://www.pyxisinc.com/NNPP_Article.pdf* (Accessed on 17-Nov-2008).
6. G.P. Sudhakar and G.Manjula, "Protecting the Programmer's Productivity", Businessgyan, October 2004.
7. John Bennett, Jr. (2006), "Career Paths for Programmers", *developer.**, August 21, 2006.
8. Robert H. Dun (1990), *Software Quality: Concepts and Plans*, Prentice Hall, Englewood Cliffs, 1990.
9. Suresh Malladi (2008), "Measuring the Software Engineering Productivity: Accounting for the Qualitative Aspects and Pitfalls", *Projects and Profits*, February 2008.
10. Warren Harrison, "Skinner Wasn't a Software Engineer", *IEEE Software*, May/June 2005.
11. *http://www.sei.cmu.edu*